

COMMUNICATIONS

CACM.ACM.ORG

OF THE

ACM

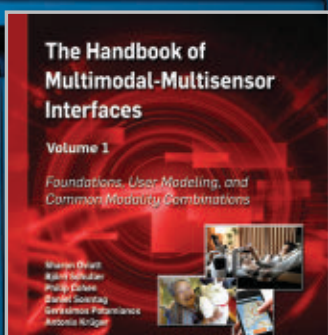
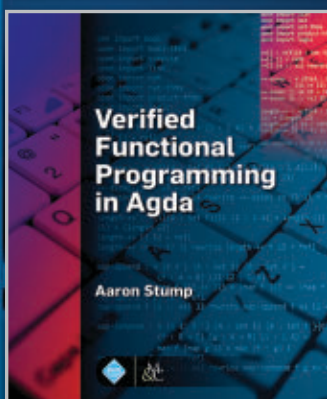
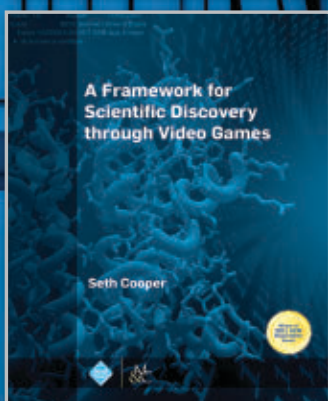
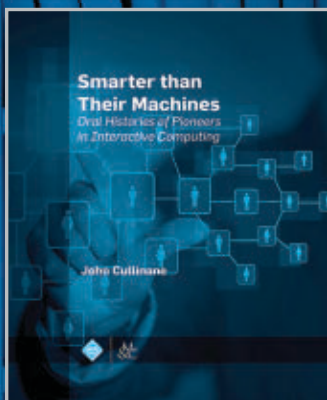
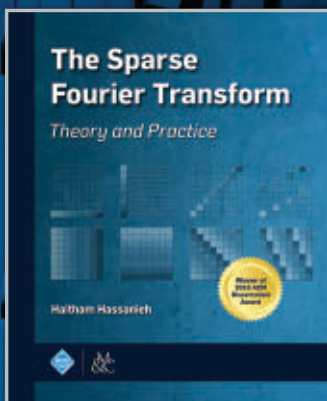
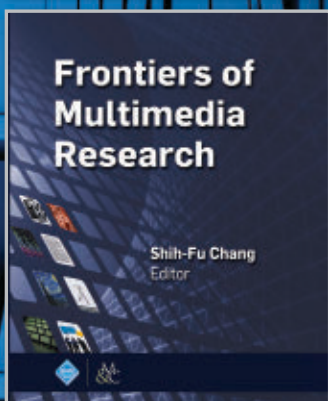
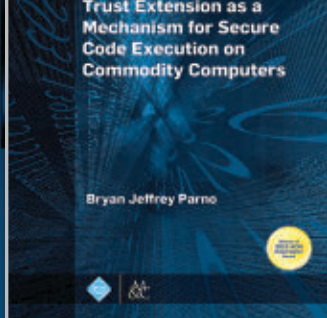
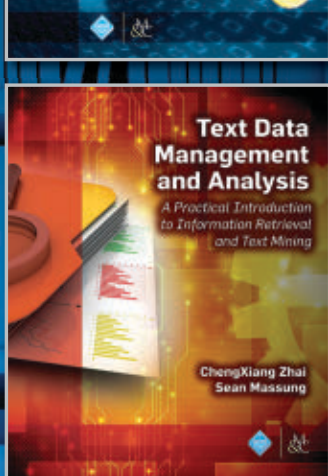
06/2023 VOL.66 NO.06



Bob Metcalfe

Recipient of ACM's A.M. Turing Award





In-depth. Innovative. Insightful.

Inspired by the need for high-quality computer science publishing at the graduate, faculty, and professional levels, ACM Books are affordable, current, and comprehensive in scope.

**Full Collection | Title List
Now Available**

For more information, please visit
<http://books.acm.org>



Association for Computing Machinery

1601 Broadway, 10th Floor, New York, NY 10019-7434, USA

Phone: +1-212-626-0658 Email: acmbooks-info@acm.org



ACM BOOKS

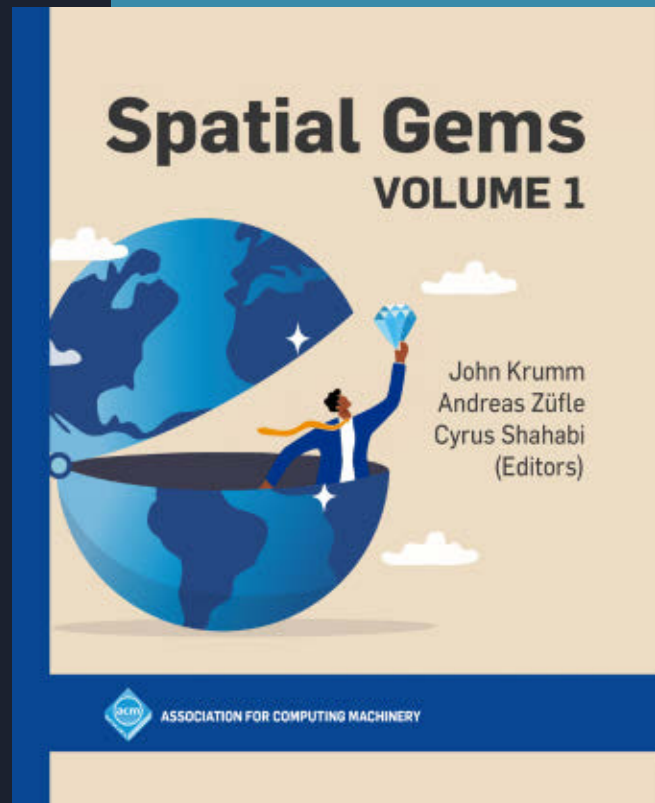
Collection II

This book presents fundamental new techniques for understanding and processing geospatial data. These “spatial gems” articulate and highlight insightful ideas that often remain unstated in graduate textbooks, and which are not the focus of research papers. They teach us how to do something useful with spatial data, in the form of algorithms, code, or equations. Unlike a research paper, *Spatial Gems, Volume 1* does not focus on “Look what we have done!” but rather shows “Look what YOU can do!” With contributions from researchers at the forefront of the field, this volume occupies a unique position in the literature by serving graduate students, professional researchers, professors, and computer developers in the field alike.

- Introduction
- Heat Map Segmentation
- Online Location Trajectory Compression
- Spatial Data Generators
- Complete and Sufficient Spatial Domination of Multidimensional Rectangles
- Minimal Representations of Polygons and Polyhedra
- Simplification of Indoor Space Footprints
- Speed Distribution from Normally Distributed Location Measurements
- A Practical Network Layout Planning System Using GeoSpatial Data
- The Brownian Bridge for Space–Time Interpolation
- A Scalable In-Memory Solution for Real-Time K Nearest Search on Road Network
- Employing GPUs to Accelerate Exact Geometric Predicates for 3D Geospatial Processing
- Using Segment Fraction for Road-Network Locations
- Flexible Computation of Multidimensional Histograms
- TINET: Triangulated Irregular Networks Evolving in Time
- Acknowledgments
- References
- Authors’ Biographies
- Index

<http://books.acm.org>

<http://store.morganclaypool.com/acm>



Spatial Gems Volume 1

**John Krumm
Andreas Züfle
Cyrus Shahabi
(Editors)**

ISBN: 978-1-4503-9811-4

DOI: 10.1145/3548732

Departments

- 5 **Cerf's Up**
Connectivity Is Everything
By Vinton G. Cerf
-
- 6 **BLOG@CACM**
AI in the Enterprise
 Doug Meil identifies the development behaviors that offer the highest chance of success for enterprise artificial intelligence efforts.
-
- 116 **Careers**

Last Byte

- 120 **Q&A**
Still Networking
 Bob Metcalfe, who co-invented Ethernet networking technology, continues to take on new challenges as a computational engineer.
By Leah Hoffmann

News



- 8 **Making Connections**
 Bob Metcalfe, recipient of the 2022 ACM A.M. Turing Award for his role in the development of Ethernet, briefly considered a career in tennis.
By Neil Savage



Watch Metcalfe discuss his work in the exclusive *Communications* video.
<https://cacm.acm.org/videos/2022-acm-turing-award>

- 11 **Better Algorithms through Faster Math**
 The search for new algorithms that could reduce the time needed for multiplication is now at the center of data science.
By Samuel Greengard
-
- 14 **Standards to Secure the Sensors That Power IoT**
 Existing security standards do not always offer sufficient protection to secure the Internet of Things.
By Logan Kugler
-
- 17 **Using Algorithms to Deliver Disaster Aid**
 Ensuring the billions of dollars governments send the disaster-stricken actually get to their intended destinations.
By Keith Kirkpatrick

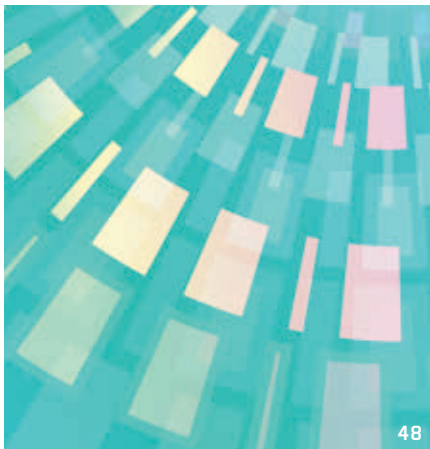
Viewpoints

- 24 **The Profession of IT**
Can Generative AI Bots Be Trusted?
 It will be a long road to learning how to use generative AI wisely.
By Peter J. Denning
-
- 28 **Inside Risks**
Computer-Related Risks and Remediation Challenges
 Surveying the nontechnical issues interwoven with computer-related technologies.
By Peter G. Neumann
-
- 31 **Kode Vicious**
The Parchment Path?
 Is there ever a time when learning is not of value—for its own sake?
By George V. Neville-Neil
-
- 33 **Historic Reflections**
Conjoined Twins: Artificial Intelligence and the Invention of Computer Science
 How artificial intelligence and computer science grew up together.
By Thomas Haigh
-
- 38 **Viewpoint**
Breaking Up a Digital Monopoly
 How to decompose a vertically integrated digital monopoly to enable competitive services based on a shared data structure.
By Micah D. Beck and Terry R. Moore
-
- 42 **Viewpoint**
Trust Is Not Enough: Accuracy, Error, Randomness, and Accountability in an Algorithmic Society
 Searching for improved algorithmic accountability.
By Emanuel Moss
-
- 45 **Viewpoint**
Decentraland: The Alleged Decentralization of Blockchain Applications
 Breaking down the blockchain.
By Massimo Franceschet



Association for Computing Machinery
 Advancing Computing as a Science & Profession

Practice



48

48 **To PiM or Not to PiM**

The case for in-memory inferencing of quantized CNNs at the edge.

By Gabriel Falcao
and João Dinis Ferreira

56 **Taking Flight with Copilot**

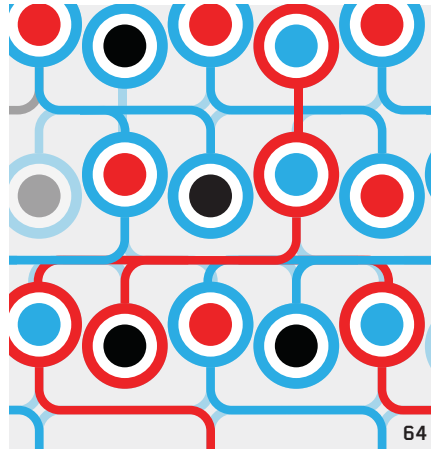
Early insights and opportunities of AI-powered pair-programming tools.

By Christian Bird, Denae Ford,
Thomas Zimmermann,
Nicole Forsgren, Eirini Kalliamvakou,
Travis Lowdermilk, and Idan Gazit



Articles' development led by **acmqueue**
queue.acm.org

Contributed Articles



64

64 **The Silent (R)evolution of SAT**

Today's powerful, robust SAT solvers have become primary tools for solving hard computational problems.

By Johannes K. Fichte, Daniel Le Berre,
Markus Hecher, and Stefan Szeider

73 **Buffer-Bloated Router?
How to Prevent It
and Improve Performance**

Home broadband routers suffering from bufferbloat can degrade the user experience when accessing the Internet.

By Philippa Harrison

Review Articles

78 **Should Robots Have Rights or Rites?**

A Confucian cross-cultural exploration of the ethical treatment of robots.

By Tae Wan Kim and Alan Strudler



Watch the authors discuss this work in the exclusive *Communications* video.
<https://cacm.acm.org/videos/robots-rights-or-rites>

86 **On the Implicit Bias in Deep-Learning Algorithms**

Examining the implicit bias in training neural networks using gradient-based methods.

By Gal Vardi

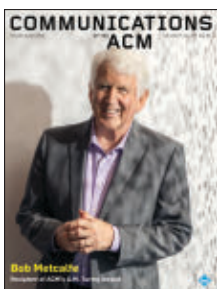
Research Highlights

96 **Technical Perspective
FoundationDB Performs
Balancing Act**

By Alfons Kemper

97 **FoundationDB: A Distributed Key-Value Store**

By Jingyu Zhou, Meng Xu,
Alexander Shraer, Bala Namasivayam,
Alex Miller, Evan Tschannen,
Steve Atherton, Andrew J. Beamon,
Rusty Sears, John Leach,
Dave Rosenthal, Xin Dong,
Will Wilson, Ben Collins,
David Scherer, Alec Grieser,
Yang Liu, Alvin Moore,
Bhaskar Muppana, Xiaoge Su,
and Vishesh Yadav

106 **Technical Perspective
What Does Provable Security
Mean for Cryptographic Schemes?**
By David Pointcheval107 **On the (In)Security of ElGamal in OpenPGP**
By Luca De Feo, Bertram Poettering,
and Alessandro Sorniotti**About the Cover:**

ACM A.M. Turing Award recipient Bob Metcalfe photographed in the Cockrell School of Engineering at the University of Texas at Austin. The award recognizes Metcalfe's contributions as co-inventor of Ethernet. Photo by Alexander Berg.



ACM, the world's largest educational and scientific computing society, delivers resources that advance computing as a science and profession. ACM provides the computing field's premier Digital Library and serves its members and the computing profession with leading-edge publications, conferences, and career resources.

Executive Director and CEO
Vicki L. Hanson
Deputy Executive Director and COO
Patricia Ryan
Director, Office of Information Systems
Wayne Graves
Director, Office of Financial Services
James Schembari
Director, Office of SIG Services
Donna Cappel
Director, Office of Publications
Scott E. Delman

ACM COUNCIL

President
Yannis Ioannidis
Vice-President
Elisa Bertino
Secretary/Treasurer
John West
Past President
Gabriele Kotsis
Chair, SGB Board
Jens Palsberg
Co-Chairs, Publications Board
Joseph Konstan and Divesh Srivastava
Members-at-Large
Nancy M. Amato; Tom Crick;
Susan Dumais; Rashmi Mohan;
Mehran Sahami; Alejandro Saucedo;
Michelle Zhou
SGB Council Representatives
Jeanna Neefe Matthews and Vivek Sarkar

BOARD CHAIRS

Education Board
Elizabeth Hawthorne and Chris Stephenson
Practitioners Board
Terry Coatta

REGIONAL COUNCIL CHAIRS

ACM Europe Council
Chris Hankin
ACM India Council
Abhiram Ranade
ACM China Council
Wenguang Chen

PUBLICATIONS BOARD

Co-Chairs
Wendy Hall and Divesh Srivastava
Board Members
Jonathan Aldrich; Tom Crick;
Jack Davidson; Chris Hankin; Mike Heroux;
Michael Kirkpatrick; Joseph Konstan;
James Larus; Marc Najork;
Holly Rushmeier; Bobby Schnabel;
Bhavani Thuraisingham;
Adelinde Uhrmacher; Philip Wadler;
Julie Williamson

DIGITAL LIBRARY BOARD

Chair
Jack Davidson
Board Members
Phoebe Ayers; Stephen Downie;
Michael Ley; Michael L. Nelson;
George Neville-Neil; Natasha Noy;
Louisa Raschid; Theo Schlossnagle;
Harald Större; Julie Williamson

COMMUNICATIONS OF THE ACM

Trusted insights for computing's leading professionals.

Communications of the ACM is the leading monthly print and online magazine for the computing and information technology fields. *Communications* is recognized as the most trusted and knowledgeable source of industry information for today's computing professional. *Communications* brings its readership in-depth coverage of emerging areas of computer science, new trends in information technology, and practical applications. Industry leaders use *Communications* as a platform to present and debate various technology implications, public policies, engineering challenges, and market trends. The prestige and unmatched reputation that *Communications of the ACM* enjoys today is built upon a 50-year commitment to high-quality editorial content and a steadfast dedication to advancing the arts, sciences, and applications of information technology.

STAFF

DIRECTOR OF PUBLICATIONS

Scott E. Delman
cacm-publisher@cacm.acm.org

Executive Editor

Diane Crawford
Managing Editor
Thomas E. Lambert

Senior Editor

Ralph Raiola
Senior Editor/News
Lawrence M. Fisher

Web Editor

David Roman
Editorial Assistant
Danbi Yu

Art Director

Andrij Borys
Associate Art Director
Margaret Gray
Assistant Art Director
Mia Angelica Balaquiot
Production Manager
Bernadette Shade
Intellectual Property Rights Coordinator
Barbara Ryan
Advertising Sales Account Manager
Ilia Rodriguez

Columnists

Michael L. Best; Michael Cusumano;
Peter J. Denning; Thomas Haigh;
Leah Hoffmann; Mari Sako;
Pamela Samuelson; Marshall Van Alstyne

CONTACT POINTS

Copyright permission
permissions@hq.acm.org
Calendar items
calendar@cacm.acm.org
Change of address
acmhelp@acm.org
Letters to the Editor
letters@cacm.acm.org

REGIONAL SPECIAL SECTIONS

Co-Chairs
Virgilio Almeida, Haibo Chen,
Jakob Rehof, and P. J. Narayanan
Board Members
Sherif G. Aly; Panagioti Fatourou;
Chris Hankin; Sue Moon; Tao Xie;
Kenjiro Taura

WEBSITE
<https://cacm.acm.org>

WEB BOARD

Chair
James Landay
Board Members
Marti Hearst; Jason I. Hong;
Wendy E. MacKay

AUTHOR GUIDELINES

<https://cacm.acm.org/about-communications/author-center>

ACM U.S. TECHNOLOGY POLICY OFFICE

Adam Eisgrau
Director of Global Policy and Public Affairs
1701 Pennsylvania Ave NW, Suite 200
Washington, DC 20006 USA
T (202) 580-6555; acmpo@acm.org

COMPUTER SCIENCE TEACHERS ASSOCIATION

Jake Baskin
Executive Director

EDITORIAL BOARD

EDITOR-IN-CHIEF

James Larus
eic@cacm.acm.org

Deputy to the Editor-in-Chief

Morgan Denlow
cacm.deputy.to.eic@gmail.com

SENIOR EDITORS

Andrew A. Chien
Moshe Y. Vardi

NEWS

Chair

Tom Conte

Board Members

Siobhán Clarke; Mei Kobayashi;
Michael R. Lyu; Rajeev Rastogi

VIEWPOINTS

Co-Chairs

Susanne E. Hambrusch and
Jeanna Matthews

Board Members

Terry Benzel; Judith Bishop;
Florence M. Chee; Vijay Chidambaram;
Danish Contractor; Lorrie Cranor;
Janice Cuny; James Grimmelmann;
Mark Guzdial; Britney Johnson;
Beng Chin Ooi; Christina Pöpper;
Alessandra Raffaetà; Chiara Renso;
Francesca Rossi; Len Shustek;
Loren Terveen; Marshall Van Alstyne;
Susan J. Winter

PRACTICE

Co-Chairs

Stephen Bourne and George Neville-Neil

Board Members

Peter Alvaro; Betsy Beyer; Terry Coatta;
Stuart Feldman; Nicole Forsgren;
Camille Fournier; Jessie Frazelle;
Benjamin Fried; Chris Grier;
Tom Killalea; Tom Limoncelli;
Kate Matsudaira; Erik Meijer;
Theo Schlossnagle; Kelly Shortridge;
Phil Vachon; Jim Waldo

CONTRIBUTED ARTICLES

Co-Chairs

m.c. schraefel and Premkumar T. Devanbu

Board Members

Pramod Bhatotia; Indrajit Bhattacharya;
Alan Bundy; Peter Buneman; Haibo Chen;
Sally Fincher; Kathi Fisler;
Jane Cleland-Huang; Rebecca Isaacs;
Trent Jaeger; Michael Jones; Gal A. Kaminka;
Ben C. Lee; David Lo; Joe Peppard;
Abhik Roychoudhury; Katie A. Siek;
Charles Sutton

RESEARCH HIGHLIGHTS

Co-Chairs

Shriram Krishnamurthi
and Orna Kupferman

Board Members

Martin Abadi; Sanjeev Arora;
Maria-Florina Balcan; Azer Bestavros;
David Brooks; Stuart K. Card; Jon Crowcroft;
Lieven Eeckhout; Gernot Heiser;
Takeo Igarashi; Nicole Immorlica;
Srinivasan Keshav; Sven Koenig; Karen Liu;
Claire Mathieu; Joanna McGrenere;
Tamer Özsu; Tim Roughgarden;
Guy Steele, Jr.; Wang-Chiew Tan;
Robert Williamson; Andreas Zeller

Association for Computing Machinery (ACM)

1601 Broadway, 10th Floor
New York, NY 10019-7434 USA
T (212) 869-7440; F (212) 869-0481

ACM Copyright Notice

Copyright © 2023 by Association for Computing Machinery, Inc. (ACM). Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or fee. Request permission to publish from permissions@hq.acm.org or fax (212) 869-0481.

For other copying of articles that carry a code at the bottom of the first or last page or screen display, copying is permitted provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center; www.copyright.com.

Subscriptions

An annual subscription cost is included in ACM member dues of \$99 (\$40 of which is allocated to a subscription to *Communications*); for students, cost is included in \$42 dues (\$20 of which is allocated to a *Communications* subscription). A nonmember annual subscription is \$269.

Single Copies

Single copies of *Communications of the ACM* are available for purchase. Please contact acmhelp@acm.org.

ACM ADVERTISING DEPARTMENT

1601 Broadway, 10th Floor
New York, NY 10019-7434 USA
T (212) 626-0686
F (212) 869-0481

Advertising Sales Account Manager

Ilia Rodriguez
ilia.rodriguez@hq.acm.org

Media Kit acmm mediasales@acm.org

COMMUNICATIONS OF THE ACM

(ISSN 0001-0782) is published monthly by ACM Media, 1601 Broadway, 10th Floor New York, NY 10019-7434 USA. Periodicals postage paid at New York, NY 10001, and other mailing offices.

POSTMASTER

Please send address changes to *Communications of the ACM* 1601 Broadway, 10th Floor New York, NY 10019-7434 USA

Printed in the USA.



Association for
Computing Machinery





DOI:10.1145/3595116

Vinton G. Cerf

Connectivity Is Everything

The Turing Award Laureate for 2022, Bob Metcalfe, coined Metcalfe's Law: The value of the network grows as the square of the number of things connected to it.

While some might quibble that the more realistic metric might be $n \log n$, the point is well made. As the inventor of the Ethernet, now ubiquitous in so many forms including its wireless origins (ALOHANET) and present form as Wi-Fi as well as wired and optical versions, Metcalfe has earned the right to speak about connectivity. His premise applies equally well to the Internet: connectivity is its essential property. When Robert Kahn and I started working on the Internet in 1973, a primary assumption and driving design premise was that anything on the Internet should be able to connect to anything else, regardless of which network the source and destination happened to be on.

Part of the rationale for this position was uncertainty about all the possible applications that might evolve for interconnected, programmable devices. The original driver for the Internet design was its anticipated service as a backbone network for command and control. This included voice, video, and data transport and interaction among processes running in the attached computers. Very quickly, the implementers of this experimental system invented networked electronic mail or email as it is now known. This proved to be remarkably beneficial to allow researchers to stay current with each other's work without both having to be awake at the same time as would be the case for phone and conference calls. Mailing lists soon followed. This computer-mediated communi-

cation opened many new possibilities for software to facilitate and augment information sharing.

The ultimate information sharing tool, the World Wide Web, was developed by Sir Tim Berners-Lee, another ACMA.M. Turing Award recipient. For that to work, every Web page needed to be able to refer to any other Web page using the ubiquitous Uniform Reference Locator (URL) construct of the Web design. Once again, connectivity is everything for the success of this massive information repository. Other aspects of the Internet's design underscore the importance of connectivity. The Domain Name System allows anyone to find by domain name the Internet Protocol address of any host that is registered in the system. The Border Gateway Protocol is a critical part of the global routing system that allows Autonomous Systems to find one another and to move traffic from a source to a destination. An interesting additional conceptual evolution is the Content Distribution

Network (CDN) that provides users with access to the "nearest" copy of content, reducing latency and avoiding repeated transfers of content across the global Internet.

An ambitious evolution of the Internet is now under way in the form of the Solar System Internet using new protocols: The Bundle Protocol Suite. These are designed to overcome the consequences of long propagation delays between the planets, thanks to the slow speed of light and disruption of connectivity owing to celestial motion. Once again, connectivity is everything. Without the ability for spacecraft and manned or robotic bases to communicate with each other and with Earth-based resources, their utility would be dramatically reduced. Although it is early days for the Solar System Internet, the space agencies of the world are actively mounting missions to return to the Moon and planning new missions to Mars and the other planets of the Solar system. In those endeavors too, connectivity will indeed be everything.

We celebrate the annual ACM Awards this month in San Francisco, CA, and I would like to extend my congratulations to all honorees. They represent the best in computer science and computer engineering. They are role models for our younger members and others interested in the science and engineering of computing. 

Vinton G. Cerf is vice president and Chief Internet Evangelist at Google. He served as ACM president from 2012–2014.

Copyright held by author.

As the inventor of the Ethernet ... Metcalfe has earned the right to speak about connectivity.

The *Communications* website, <https://cacm.acm.org>, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.



Follow us on Twitter at <http://twitter.com/blogCACM>

DOI:10.1145/3591212

<https://cacm.acm.org/blogs/blog-cacm>

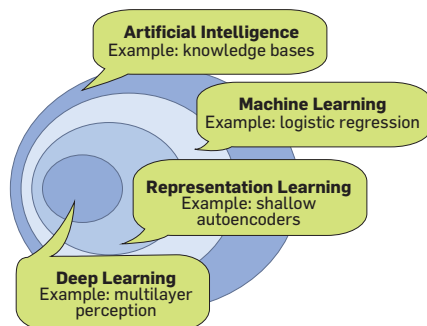
AI in the Enterprise

Doug Meil identifies the development behaviors that offer the highest chance of success for enterprise artificial intelligence efforts.



Doug Meil
Enabling AI Projects in an Enterprise
<https://bit.ly/3vRxVPB>
March 2, 2023

Most people in the technical world are now familiar with some version of this artificial intelligence (AI) Venn diagram:



describing the relationship between various sets of AI techniques, including machine learning and deep learning. There are many excellent books and articles describing those topics and how they can be implemented in various software frameworks, and those descriptions will not be repeated here. There also are many articles on Big Tech implementing AI at scale. But how do “regular” organizations implement AI projects successfully,

especially within an existing portfolio of solutions? In the BLOG@CACM post “*Anna Karenina on Development Methodologies*,” I described how the famous opening line “*happy families are all alike, unhappy families are unhappy each in their own way*” applies to software development. This post will describe in a similar vein the development behaviors with the highest chance of success for AI efforts.

The AI Project Bootstrapping Dilemma

Pick a use case, or at least, pick a few candidate problem/opportunities to research. This is, “they say,” advice that is obvious but necessary to repeat, because it is true. There is a marketing tendency to want to sprinkle AI on everything and see what grows, with the hope that something magical will happen. IBM’s Watson AI business was notorious for this when it referred to its namesake AI framework as “Cognitive Computing” back in the Watson Health era, a phrase which implied a great many things but meant nothing specific and had the effect of inflating customer expectations to an unmanageable degree.

It is also easy to get blinded in a myriad of potential AI technical implementation issues and lose sight of the

analytic goals for what was originally supposed to be clustered, categorized, or predicted. The adage about alligators and draining the swamp applies.

Multilevel alignment. In the BLOG@CACM post “*Developing Technical Leaders*,” I described common levels of leadership progression in software engineering, ranging from individual contributor to tech-lead/senior individual contributor to team lead to manager. AI efforts are a prime example of the necessity of multilevel alignment, because any successful effort needs things such as:

- ▶ A technical person who can do the work.
- ▶ A technical person who can get the project approved.

These abilities rarely exist in the same person. Sometimes in an organization, there is a staff member with an idea but with no ability to get it prioritized, and sometimes there might be leaders with a general idea for an AI effort, but with no ability to execute.

Multilevel alignment also applies to use case selection as well, as there can be a difference between “executive understanding” of use cases and those that experience pain-points in person. Both are valuable perspectives, but they are distinct. This represents the second set of multilevel alignment for stakeholders:

- ▶ Someone who can speak to the use case with specificity.
- ▶ Someone who can advocate for the use case and get it prioritized.

As on the technical side of the house, those people are rarely the same.

Establish an AI baseline. Per the Venn diagram here, it is important everyone on the project team is using the same vocabulary and can explain things such as the difference between unsupervised and supervised learning, the difference between a classifier and regression, and common steps in required data preparation. This includes the stakeholders because understanding these concepts as much as possible is essential for expectation management, as these are not just “implementation details,” it is about understanding the actual art of the possible, and what is realistic.

As the outer circle of concepts that “AI” contains is quite large, start with the basics and go from there.

Data investigation. Data is where the rubber hits the road with AI projects, and it is why having individual contributors who understand both the relevant technology and problem space is so critical for effective data research. Never take for granted the following questions:

- For the problem being researched, is there enough data available?
- For the data that exists, how clean is it?
- What data governance is relevant for the context?

A project team might be instinctively digging into an actual bonafide problem area only to find that there is not enough data to support the proposed analytics, or at least analytics with desired outcomes. Likewise, even when data exists, the reason why people repeat the quip “80% of data science is data preparation” is because it is true—so, so painfully true. Lastly, understanding relevant data governance is a requirement especially in regulated industries, such as healthcare. “Why can’t you use test or fake data for this project?” is a common compliance response with respect to both AI and natural language processing (NLP) efforts. While fake data is suitable for basic solution development in terms of having characters to display on a page or basic report, when trying to predict something in the real world one generally needs real data, or something very closely approximating the real thing—the generation of which is often an even more difficult problem than just using real data it-

The more critical the use case, the more real and comprehensive the training data needs to be.

self. The more critical the use case, the more real and comprehensive the training data needs to be. AI and NLP efforts tend to raise non-trivial governance questions, and my BLOG@CACM post “Data Governance and the Psychology of Tension Management” describes some of the challenges where data needs to be protected, but also needs to be utilized to be useful.

The power of feedback. Flying a plane is an activity that, on the surface, looks sort of easy when conditions are perfect, until they are not, and then things get complicated in a hurry. But it is not just about the plane, as Malcolm Gladwell described in his book *Outliers* (<https://amzn.to/412eodU>) when discussing the concept of Power Distance and mitigation in communication. How the crew *talks* about solving problems is arguably even more important than the underlying problems themselves, as the communication approach plays a critical role not just in managing a crisis, but also preventing the further compounding of errors.

There are parallels in software development. Let’s say an AI use case was selected at the executive ranks. Even if the use case was chosen with the best of intentions, the use case might not be ready for AI for some of the reasons already stated. Now what? The best situation would be for all (including the executive team) to learn and understand what was impeding that AI use case and try to remedy, and the remedies may be beyond what the project team can do by themselves—especially if changes are required in a source system. Actions on the other end of the spectrum would be for the project team to continue to bang

their collective heads on the original use case instead of pivoting to research another one, either for fear of “being wrong” or fear of delivering bad news up the chain. High-quality AI is not typically something that can be summoned by executive decree.

Operationalize. It is natural to start with the prototypes or pilots as stand-alone efforts, but eventually the output needs to be integrated with the rest of the solution portfolio for maximum organizational benefit. The “last analytic mile” can be a difficult one as in healthcare settings, for example, providers spend an inordinate amount of time in electronic medical record systems looking at, and entering, patient data. Asking providers to log into a second, third, or fourth set of solutions to see patient analytic results adds to the workload, especially when seeing 15, 20, or more patients a day. There might be some amazing supplemental AI analytic dashboards in the portfolio, but there also might not be enough time to view them in the workflow when extra context switching is required.

Conclusion

My “Anna Karenina” post described velocity as a foundational attribute in software development. Frequent software releases cannot guarantee success, but sporadic releases and inconsistent funding have doomed more products and solutions to failure than any other reason. Similarly, early AI results could be a bit rough in the beginning. Keep learning and keep iterating. C

References

BLOG@CACM

1. Developing Technical Leaders: <https://bit.ly/3x4aJiv>
2. Data Governance And The Psychology Of Tension Management: <http://bit.ly/3Mz6Mfh>
3. Anna Karenina On Development Methodologies: <https://bit.ly/3vRxVPB>
4. What Happened To Watson Health?: <https://bit.ly/3v4sawP>

Power Distance

5. Malcolm Gladwell On Plane Crashes: <http://bit.ly/3m9qgfH>

Doug Meil is a software architect in healthcare data management and analytics. He also founded the Cleveland Big Data Meetup in 2010. More of his BLOG@CACM posts can be found at <https://www.linkedin.com/pulse/publications-doug-meil>.

Making Connections

Bob Metcalfe, recipient of the 2022 ACM A.M. Turing Award for his role in the development of Ethernet, briefly considered a career in tennis.

WHEN HE WAS a student at the Massachusetts Institute of Technology (MIT), Ethernet inventor Bob Metcalfe briefly considered pursuing a career in tennis. He was captain of the 1968–1969 MIT tennis team, which had a record of 15 wins and 4 losses, and he was ranked sixth in New England in doubles, even while taking classes and holding a programming job at defense contractor Raytheon. That, unfortunately, was not enough to make a go of it.

“There’s playing pros and there’s teaching pros,” Metcalfe says. “I could easily be a teaching pro, but that just seemed boring. And for being a playing pro, I wasn’t good enough.”

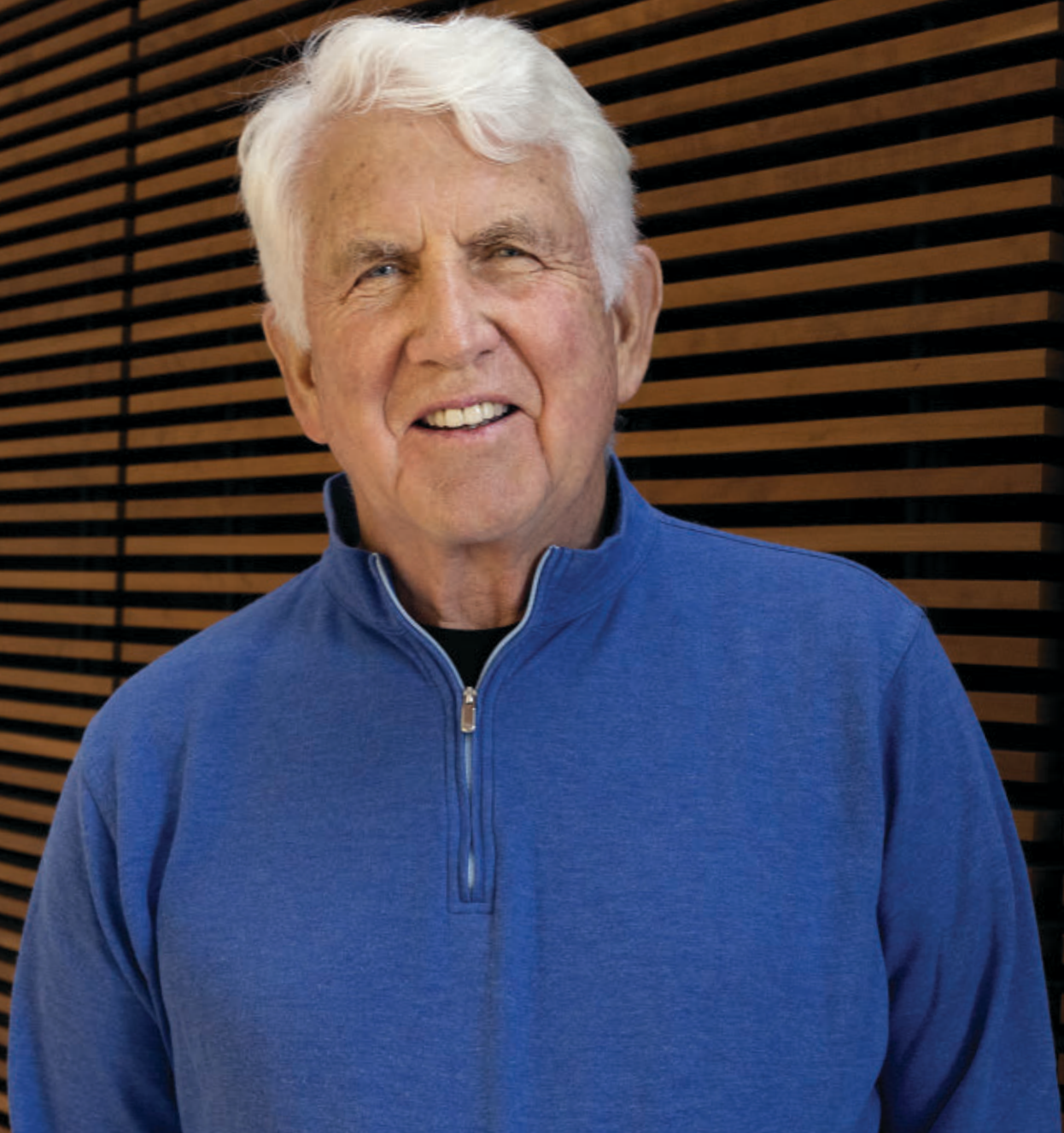
Metcalfe wrote his undergraduate thesis on a bus coming back from a tennis match and submitted it to Minsky at the last possible moment.

The tennis world’s loss was the computer world’s gain, however, as Metcalfe went on to become an Internet pioneer, develop Ethernet, and help get it named a networking standard, actions that earned him the 2022 ACM A.M. Turing Award on the 50th anniversary of the invention of the technology.

Metcalfe was working at Xerox’s Palo Alto Research Center (PARC) in 1973, having just earned his Ph.D. from Harvard University. At that time, the standard model for computer users was to have dumb terminals with no processing power at their desks, all linked to a central computer. PARC scientists decided to build a personal computer so users could run software at their desks, but needed a way to network the machines. At the same time, Xerox was building laser printers, which could handle input at a rate of 20Mb/s; the dumb terminal network operated at only 300b/s.

Metcalfe teamed up with electrical engineer David Boggs, who died last year, to come up with a way to build a fast, reliable local area network. Their only constraint was that it had to fit on a card that held 60 medium-scale integration (MSI) chips, each of which held hundreds of transistors, a much smaller number than today’s computer chips.

The pair set out to design a system that could use as few wires as possible—preferably zero. At the time, though, radios were too bulky to make wireless networking feasible, so they settled on one connection, a coaxial cable. They chose the name “Ethernet” to represent the idea that the system would be agnostic about the medium that carried the signal. In the 19th century, physicists had believed that light traveled through a medium they called “the luminiferous ether,” but in the early 20th century the Michelson-Morley experiment proved that no such thing existed. That left the term “ether” up for grabs, and Metcalfe grabbed it.



The original Ethernet worked on three basic technologies, none of which are still in use. One was the Jerrold tap, a method used by the cable television industry to tap into a coaxial cable. The second was Manchester encoding, in which each bit cell was either 1-0 or 0-1 transition. That had the advantage of embedding a clock into each data packet. Boggs and Metcalfe still needed a way to get nodes of the network to take turns retransmitting packets in the event two of them interfered with each other.

Metcalfe's friend Steve Crocker suggested he look at the retransmission scheme developed by ALOHAnet, a radio-based network created by the

University of Hawaii to connect computers on different Hawaiian islands. Metcalfe spent a month in Hawaii studying their system and determined that randomized retransmission was the right method to use when modified for cable.

A Deep Background

Metcalfe had long been interested in technology. His father was a gyroscope technician and had a basement room filled with electronic junk that the two of them used to build a train set. In 1959, Metcalfe's eighth-grade teacher encouraged him to create something out of the material, so he made a rudimentary computer capable of add-

ing any number from 1 to 3 to another number from 1 to 3. "I built it out of relays, toggle switches, and neon lights," he recalls.

A few years later he went to MIT to study electronic engineering, which was what the school called 'computer science' in those days. His advisor was Marvin Minsky, who pioneered the field of artificial intelligence (AI). "Marvin Minsky didn't think I was smart enough to do AI, so I went into networking instead, and it worked out wonderfully," he says. "So, thank you, Marvin." Metcalfe wrote his undergraduate thesis on a bus coming back from a tennis match and submitted it to Minsky at the last possible moment.

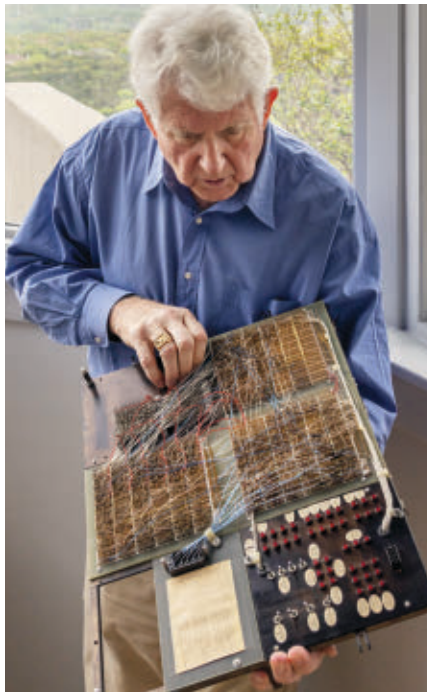
After leaving MIT in 1969, he went to Harvard to earn a master's degree and a Ph.D. in applied mathematics. He focused his research on then-new ARPANET, the first public, packet-switched network, created by the U.S. Department of Defense's Advanced Research Projects Agency. He offered to build the interface that would connect Harvard to ARPANET, but "Harvard said no. 'This interface between the Harvard and the ARPANET is too important for a grad student'," he said he was told. Instead, Metcalfe was hired by MIT to build the interface.

In 1979, he left Xerox PARC and founded 3Com, which made Internet hardware and software. He also worked as a consultant for Digital Equipment Corporation (DEC), then the second-largest computer maker after IBM. DEC's vice president of engineering, Gordon Bell, asked him to design a networking protocol for that company. Metcalfe recalled, "I told Gordon that I wouldn't do that for him, both because I felt some sort of loyalty to Xerox, but also I felt that Ethernet was already the best one I could come up with."

Instead, Metcalfe and Bell decided in February 1979 that Xerox and DEC should combine their technologies so Xerox printers could work with DEC computers. Metcalfe ran into an Intel engineer looking for a standard to apply to their chip technology, so he got the three companies together, aiming to make Ethernet a standard. "I was the marriage broker—that's their term—that brought DEC, Intel, and Xerox (DIX) together," Metcalfe says.

To satisfy anti-trust laws, DIX pushed for an open standard, IEEE Project 802. Soon, IBM and General Motors showed up to their meetings, each pushing their own standards. The IEEE decided to standardize all three proposals, but only Ethernet remains. Bell says that is because it proved superior to other approaches. "I have to put Ethernet as probably one of the most elegant designs I know of," he says, which is why it has become essentially the core of the modern Internet.

As for Metcalfe, Bell says, "Bob is a joy to work with. Very good sense of humor and a really good view of the world."



Before inventing Ethernet, Metcalfe built this Interface Message Processor (IMP) board at MIT to connect a PDP-10 host computer to ARPANET.

Part of the way Ethernet built that industry was by allowing for much higher transmission speeds. The original Ethernet carried 2.94Mb/s, 10,000 times faster than the dumb terminal network it replaced. Ethernet has since grown to far higher speeds, allowing video to dominate the Internet.

Six Careers

Metcalfe is not one to stick with a single job forever. "When you've been at something for 10 years, it's time to move on to something new," he says. After 3Com, where he was CEO for a time and also had "a bunch of jobs" until he left in 1989, he went into journalism, publishing *InfoWorld* and writing a weekly column.

In 2001, he became a venture capitalist and a partner at the VC firm Polaris Partners. His interest in entrepreneurship was not new, though. As an undergraduate at MIT, he had spent a fifth year earning a degree in industrial management from the Sloan School of Management, during which time he founded three companies.

Ten years later, spurred by the desire of his wife, Robyn Shotwell Metcalfe, to get away from the cold New England winters, he became a professor of in-

novation and entrepreneurship at the University of Texas, Austin, where he worked to make Austin "a better Silicon Valley."

Now he is embarking on his sixth career, as a research affiliate in computational engineering at MIT, where he's working on modeling the configuration of geothermal wells for energy. "It's different, but it's also hard because I have to learn all this stuff that I haven't really learned over the years I've been doing other things, like thermodynamics" he says.

Metcalfe's advice to young people: Not everyone should be an entrepreneur, but if you want to do it, go on and do it. "It's a very high calling. It's how innovation gets done," he says. He advises against dropping out of school, and suggests people go work for someone else first. A first job can serve as a sort of post-doctoral fellowship, he says.

Last year, Metcalfe endowed a professorship of entrepreneurship at MIT Sloan. The foundation he and his wife created also has funded a writing professorship and an Internet professorship at MIT.

While he does not yet know what he will do with the \$1 million prize the comes with the Turing Award, Metcalfe has other goals. For one, he would like to "fix" science. Among his proposals are reforming the peer review system, eliminating tenure and accreditation, and dissolving the "silos" that separate scientific disciplines, which he says create a lot of unnecessary friction.

While he may have given up a career in tennis, he points out that because MIT did not have a football team at the time he was a student, at 77 he still has four years of NCAA eligibility in that sport. "You know, I weigh 220 pounds, and I got wheels and hands, but I've yet to attract the attention of the Texas football coach. Hook 'em, Horns!" 

Neil Savage is a science and technology writer based in Lowell, MA, USA.

© 2023 ACM 0001-0782/23/6 \$15.00



Watch Metcalfe discuss his work in the exclusive *Communications* video. <https://cacm.acm.org/videos/2022-acm-turing-award>

Better Algorithms through Faster Math

The search for new algorithms that could reduce the time needed for multiplication is now at the center of data science.

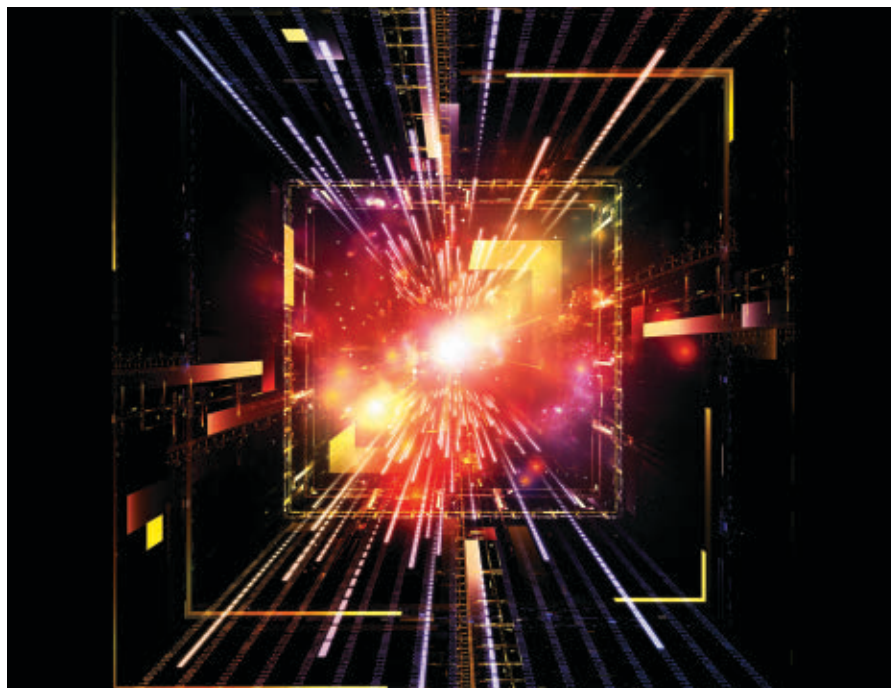
DEVELOPING FASTER ALGORITHMS is an important but elusive goal for data scientists. The ability to accelerate complex computing tasks and reduce latency has far-reaching ramifications in areas such as natural language processing, video streaming, autonomous robotics, gaming, and extended reality.

Yet for all the hype surrounding computer algorithms and the increasingly sophisticated ways they operate, a basic fact stands out: these algorithms are typically built atop matrix multiplication, a basic type of linear algebra. The underlying mathematical framework has not changed a great deal since the inception of computing—and finding more efficient formulas has proved elusive.

It is an issue attracting growing attention—particularly as machine learning (ML), deep learning (DL), artificial intelligence (AI), and machine automation advance into the mainstream. Milliseconds can make an enormous difference in terms of how well an algorithm operates, the performance it produces, and how many computing cycles it consumes.

So, not surprisingly, the search for new algorithms that could reduce multiplication is at the center of data science. Although researchers have explored this issue for many years and experimented with different techniques, in October 2022 a team from Alphabet Inc.’s U.K.-based subsidiary DeepMind reported it had found a way to uncover better algorithms through its AlphaTensor reinforcement learning framework.

Using ML, DeepMind found a way to discover novel fast matrix multiplication algorithms, says Alhussein Fawzi, staff research scientist at DeepMind and lead author of an academic



paper that appeared in *Nature*.^a While the results are more theoretical than practical at this point, the discovery represents a breakthrough. “This approach could eliminate many bottlenecks,” states Tammy Kolda, an independent consultant for MathSci.AI and a distinguished visiting professor at Northwestern University.

The research community has taken notice. While many questions remain—particularly surrounding the black-box nature of the research and how much computational power and financial resources used DeepMind—it is clear faster algorithms would have a profound impact on wide-ranging areas of AI, including climate change models, speech recognition systems, autonomous vehicles, neural networks, and graphics.

^a “Discovering faster matrix multiplication algorithms with reinforcement learning”; <https://go.nature.com/3KDEWF4>

“This extends explosive progress achieved recently in reinforcement learning,” says Victor Y. Pan, Distinguished Professor in the departments of Mathematics and Computer Science at City College of New York (CUNY) and a pioneer in the field of matrix multiplication algorithms.

Making Algorithms Add Up

Although it is tempting to view algorithms through the filter of modern data science and computing, mathematicians have used matrix multiplication methods for hundreds of years to address technical challenges in fields as diverse as finance, engineering, chemistry, and physics. Matrix multiplication, while one of the most basic operations in algebra, is also among the most powerful and utilitarian.

Today, algorithms handle a mind-bending array of tasks on various types of computing devices. They recognize speech commands, provide live navi-

gation capabilities, generate computer graphics, deliver ads, and run elaborate simulations, including Digital Twins. In fact, matrix multiplication is so important that computer hardware is specifically optimized to use it. Until 1969, the math consisted of establishing a side-by-side matrix and then multiplying the rows of one with the columns of the other. At that point, German mathematician Volker Strassen found a shortcut that trimmed the number of steps in a two-row and two-column model from 8 to 7.^b

Since then, a point of intrigue for data scientists has been whether more efficient algorithms are possible—particularly because most dense matrix multiplication algorithms use non-Strassen methods. The question is rooted in a basic fact: Every step in a multiplication table results in slower algorithm performance and increased compute cycles. Yet, research in the space has slogged forward. “Over the years, we have periodically seen small and incremental performance improvements—and many of these gains have been strictly theoretical,” says Grey Ballard, an associate professor of computing science at Wake Forest University.

Indeed, the DeepMind group rocked the data science world with its announcement. Building on AlphaZero—a Google framework that previously was used to master chess, Go, and other games through an earlier iteration known as AlphaGo—their new ML approach (called AlphaTensor) found there are more efficient ways to multiply grids. AlphaTensor cycles through matrices and generates code that can be used to multiply any matrix. What’s more, researchers can train it to search for efficient algorithms tailored to specific hardware.

The results were impressive. For example, AlphaTensor discovered an algorithm for multiplying four-by-four matrices in 47 steps, down from the 49 steps that are used in the standard Strassen algorithm. DeepMind also found it could optimize algorithms to run up to 20% faster on specific hardware, such as Nvidia V100 GPUs and

the Google TPU v2 platform. “This isn’t the same as embedding these algorithms on real world systems, but it was a significant proof of concept,” Fawzi points out.

Into the Matrix

At the outset, AlphaTensor had zero knowledge of the specific problem. DeepMind researchers set out to solve—specifically, finding shortcuts for matrix multiplication. Relying on the same general reinforcement learning methods that researchers had previously used to teach AlphaGo chess and other games, the DeepMind team constructed a single-player game and directed AlphaTensor to find highly efficient ways to win.

However, the approach differed from past projects in an important way. It eschewed techniques used to teach older AI models games through alpha-beta pruning techniques and constant human oversight and heuristics. AlphaTensor teaches itself by cycling through scenarios using a reward function. As it succeeds, it updates its policies and values. The net effect was an ability to discover new algorithms.

“An important contribution of AlphaTensor is in the formulation process itself: How can we express a problem that, at first sight, is far removed from the usual settings in machine learning as a reinforcement learning problem,” Fawzi explains. These include factors that are not part of a typical game, including “a huge action space, a highly symmetric problem, and a lack of a large dataset based on expert human demonstrations,” he notes.

Along the way, DeepMind encoun-

tered a steep challenge: the sheer size of the search space. AlphaTensor had to sort through the possibility of more algorithms than the entire number of atoms in the universe. In fact, at one point, the system had to sift through upward of 10^{33} possible steps within the game. In order to achieve meaningful results, “Machine learning had to inform the search procedure to find the interesting and useful algorithms, which would not be possible to discover with uninformed search,” Fawzi explains.

Researchers used several heuristic techniques to tackle the space problem, Fawzi explains. First, they turned to a sampled tree search, which relied on modified versions of AlphaZero, the underlying model for AlphaTensor. This allowed the group to sample actions that were more likely to produce meaningful outcomes. They also used synthetic data to show the agent techniques it could imitate, and they directed AlphaTensor to solve all matrix sizes and fields simultaneously. “This made it possible to accelerate knowledge transfer within the system,” Fawzi says.

The researchers also reformulated algorithm discovery in numerous equivalent ways, through a technique known as *change of basis*. “Once the agent finds a good solution in any of the ‘basis,’ this solution can transfer to all other instances of the problem,” Fawzi says. Although AlphaTensor still has a way to go to learn algebraic geometry, representation theory, and other complex mathematical tasks, it already has demonstrated an ability to find the proverbial “needle in a haystack.”

AI by the Numbers

DeepMind’s work has pushed matrix multiplication into new territory—and other DL agents using similar techniques may advance the field further. Ballard, who has conducted research in the same space, says the DeepMind breakthrough slides the algorithmic improvement dial to a more practical level suited for realistic applications. “When you are looking at a neural network with many layers and a large amount of data movement, the bottlenecks start to add up. Any speed increase can result in significant gains in the end result,” he says.

DeepMind researchers using AlphaTensor found there are more efficient ways to multiply grids.

^b “Gaussian elimination is not optimal,” <https://link.springer.com/article/10.1007/BF02165411>

Kolda likens the DeepMind breakthrough to finding a faster way to navigate between two cities without an existing map.

Kolda, who also has studied Tensor Decomposition and other methods to improve algorithm performance—including during a stint at Sandia National Laboratories—likens the DeepMind breakthrough to finding a faster way to navigate between two cities without an existing map. A ML model must explore every road, highway, and byway until it finds the shortest or best route. “The focus isn’t so much on how you got there, it’s that you have found a way to optimize the path.”

Yet the DeepMind methods also raise important questions about the autonomy of AI and the black box approach. For one thing, AI algorithms require less and less engineering and input from humans. “It’s a messy way to get to an algorithm that you can’t actually verify,” Kolda says. For another, “We don’t know how much computational power went into solving this problem and whether other entities have the deep pockets to make this type of research possible—or feasible on any realistic scale,” she adds.

The latter is not an abstract issue, or an obstacle that is easily addressed. In order to achieve significant gains in matrix multiplication, Fawzi admits, researchers would need to study much larger tensors, which would make the search space even larger. Essentially, the needle stays the same, but the haystack grows larger. This challenge may necessitate the use of a hybrid method that combines novel ML approaches that do not incorporate any problem-specific knowledge with more traditional methods built atop mathematical techniques. “This may help the model scale better,” he says.

In fact, Fawzi believes many computational problems can be framed as tensor decompositions. In practical terms, this means it is possible to reuse and recycle many of the techniques the DeepMind team developed with AlphaTensor. Over time, this approach could lead to significant improvements in existing algorithms, but also produce new types of algorithms for improving things such as energy consumption and numerical stability. Says Fawzi, “We believe that the solutions we built for this project will be applicable—or at least provide intuition—for other problems that extend beyond matrix multiplication.” 

Further Reading

Fawzi, A. et al.

Discovering Faster Matrix Multiplication Algorithms with Reinforcement Learning. *Nature*, 610, pages 47–53, October 22, 2022. <https://www.nature.com/articles/s41586-022-05172-4>

Discovering novel algorithms with AlphaTensor. DeepMind, October 5, 2022. <https://www.deepmind.com/blog/discovering-novel-algorithms-with-alphatensor>

Karstadt, E. and Schwartz, O.

Matrix Multiplication, a Little Faster. *J. Communications* Volume 67, Article 1. January 2020, Pages 1–31. <https://doi.org/10.1145/3364504>

Ballard, G., Demmel, J., Holtz, O., and Schwartz, O.

Communication Costs of Strassen’s Matrix Multiplication. *Communications*, Volume 57, Issue 2, February 2014, pp 107–114. <https://dl.acm.org/doi/10.1145/2556647.2556660>

Benson, A.R. and Ballard, G.

A framework for practical parallel fast matrix multiplication. *ACM SIGPLAN Notices*, Volume 50, Issue 8, August 2015, pp 42–53. <https://dl.acm.org/doi/10.1145/2858788.2688513>

Pan, V.Y.

Strassen’s Algorithm Is Not Optimal: Trilinear Technique of Aggregating for Fast Matrix Multiplication, *Proc. of the 19th Annual IEEE Symposium on Foundations of Computer Science (FOCS’78)*, 166–176, IEEE Computer Society Press, Long Beach, California, 1978 (Journal version in “New Fast Algorithms for Matrix Operations”, *SIAM J. on Computing*, 9, 2, 321–342 (1980)).

Samuel Greengard is an author and journalist based in West Linn, OR, USA.

© 2023 ACM 0001-0782/23/6 \$15.00

ACM Member News

VALUE-SENSITIVE DESIGN



Batya Friedman, is a professor at The Information School of the University of Washington

(UW), is also co-director of both the Value Sensitive Design Lab and the UW Tech Policy Lab.

Friedman earned her undergraduate degree in computer science, and her Ph.D. in Group in Science and Mathematics Education, both from the University of California, Berkeley.

On obtaining her doctorate in 1988, Friedman joined the faculty of Mills College in Oakland, CA, then moved to Colby College in Waterville, ME, before joining the faculty at the University of Washington in 1999, where she has remained.

At Colby, Friedman built the first computer science major program, as well as the Department of Computer Science. At UW, she designed and developed the undergraduate major in informatics.

Friedman pioneered value-sensitive design (VSD) in the 1990s. VSD seeks to provide theory and method to account for human values in a principled and systematic manner throughout the design of technical systems. “Value-sensitive design is an approach for foregrounding human values in the technical design process,” Friedman says.

A critical aspect of Friedman’s research focus is on the longer-term implications of current computing and information system designs. She cites earbuds as an example, and how frequently they are replaced, and the associated waste involved.

“We need to design with the constraints of the planet in mind, and re-envision our computing within those constraints,” Friedman says. She said she hopes to see the widespread adoption of VSD alongside current technical processes in the future.

—John Delaney

Standards to Secure the Sensors That Power IoT

Existing security standards do not always offer sufficient protection to secure the Internet of Things.

THE USE OF Internet of Things (IoT) sensors has exploded in popularity in recent years as cheap, effective IoT sensors make it possible to connect devices that do everything from regulating smart home features to monitoring health and fitness using wearable devices.

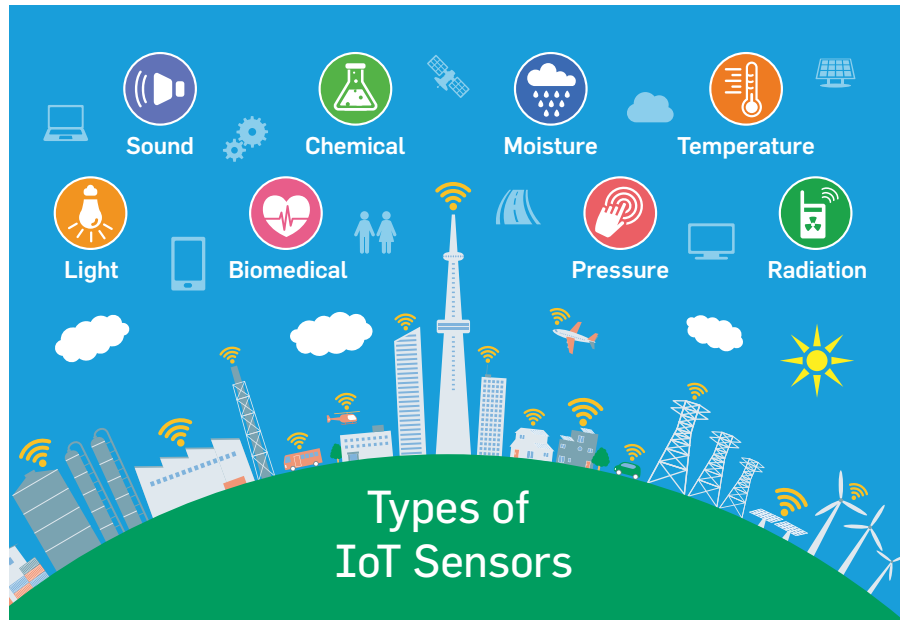
IoT sensors also are increasingly making their way into business use-cases. In the industrial IoT, sensors are used in many different contexts, including to control and monitor machinery and to regulate core infrastructure systems.

IoT device and sensor usage has accelerated even more with advances in 5G connectivity and the shift to remote work, says Willi Nelson, chief information security officer for Operational Technologies at Fortinet, a cybersecurity firm. In fact, the number of IoT devices in use is projected to nearly triple to 29 billion in 2030 compared to 9.7 billion today, according to data from Statista.

Yet as IoT adoption increases, IoT sensors and devices also are becoming more popular targets for cybercriminals.

“They remain a prime target of cybercriminals as a fast path to gain access to enterprise networks,” says Nelson. Fortinet found 93% of companies using IoT sensors in some capacity had one or more cybersecurity intrusions in the past year. A full 78% had experienced three or more, and these attacks increasingly are targeting industrial IoT operations, too.

That is because IoT is a fundamentally different technology than existing systems—a technology with plenty of attack surfaces. Each sensor and device connected to an IoT network presents a possible security risk, opening up an attack vector into an individual or company’s hardware, software, and/or data.



In theory, IoT security standards are supposed to mitigate cybersecurity risks by encouraging companies to follow best security practices when designing and deploying IoT sensors and devices.

However, in practice, the standards available to manufacturers and companies using IoT technology do not always offer sufficient protection, are not always designed specifically for IoT, and are not always followed.

Despite the vulnerability of IoT devices, quite shockingly, there is no single standard for IoT security.

No Standard Set of Standards

IoT sensors carry a variety of unique risks because they are connected to larger sensitive networks. Medical IoT devices handle sensitive and often legally protected patient and hospital data. Industrial IoT sensors connect to other critical manufacturing equipment. IoT sensors in energy offer a gateway into critical private and public power infrastructures.

One prominent example is the damage caused by the malware named “Mirai” in 2016. Mirai infected computers and devices, which in turn targeted IoT devices and sensors. Once infected, IoT devices were used to temporarily take down many popular websites, including Twitter, Netflix, and Airbnb.

Standards Are Absolutely Essential for Security

In IoT, a standard is a document providing rules and guidelines for some aspect of an IoT device’s manufacturing, functioning, and/or usage. These standards are developed by dozens of highly credible organizations within the industry, including major players such as the International Organization for Standardization (ISO), the International Electrotechnical Commission (IEC), the European Telecommunications Standards Institute (ETSI), and the Institute of Electrical and Electronic Engineers (IEEE).

By following established, credible security standards, companies mak-

ing and using IoT sensors can have a high degree of confidence their technology uses the best security features and practices.

There are plenty of security standards that IoT devices can—or should—follow. Some are related to how IoT devices use networks and transmit data. Some are related to the underlying technologies IoT devices rely upon (such as Wi-Fi). Others offer comprehensive guidance on how to create and use IoT devices in a secure way.

One well-known IoT standard is ISO/IEC 30141. According to the ISO, 30141 “provides a standardized IoT Reference Architecture using a common vocabulary, reusable designs, and industry best practices.”

Another standard, TS 103645 from ETSI, a European-focused organization, aims to create a security baseline for Web-connected devices, including guidelines for password usage, software updates, and user data standards for consumer IoT devices.

In another example, the U.S. National Institute of Standards and Technology (NIST) has created a list of six prescribed security characteristics that manufacturers should incorporate into IoT devices. The list includes security features such as device identification, device configuration features, data protection features, logical access to interfaces, adequate software and firmware updates, and adequate cybersecurity event logging.

There are dozens of organizations that publish helpful standards to guide IoT manufacturers and device customers on how to design, manufacture, and use IoT sensors and sensor-enabled devices in the safest way possible. However, the diversity of organizations and standards also presents problems.

Some standards organizations may aim to publish universal standards across different IoT technologies, while others may only publish standards for certain countries or devices and technologies. While these organizations are usually highly credible and undergo rigorous processes to ensure their standards are comprehensive, many such standards are not legally binding.

This means there is no single standard out there for IoT security.

In addition, existing standards are not always designed for the unique

Existing standards are not always designed for the unique risks that IoT technologies face.

risks IoT technologies face, says Izzat Alsmadi, a computer science professor at Texas A&M University in San Antonio, who does work on IoT security. Existing standards may not adequately apply to significant numbers of IoT sensors, he explained, and some IoT devices and networks use proprietary technology that does not follow more widely accepted or used industry standards.

“Today’s IoT standards are relevant, but not enough and in some cases not up to date or not up to security challenges,” says Alsmadi. That’s because some of today’s existing security mechanisms were initially designed for desktop computers and are difficult to implement on resource-constrained IoT devices, he says.

There also is the problem of compliance. Standards are often voluntary—and many companies do not adhere to them due to business pressures.

“Currently, the IoT segment sacrifices security due to resource allocation and price,” says Marion Marincat, founder and CEO of Mumbli, an IoT company. It is often faster and cheaper to limit security options in order to get to market, he says. As such, the standards for IoT mainly end up being adopted by the companies with deep-enough pockets and wide-enough competitive moats to afford to implement better security in their devices.

“Although there are a lot of methods to design low-cost devices with security in mind, business decisions usually push back the implementation for these solutions in order to speed up the route to market or reduce the price of devices even further,” says Marincat.

A Vulnerable Internet of Things

The issues with IoT sensor standards have larger implications for the overall

security of the Internet of Things.

“The Internet of Things is very vulnerable in comparison with other categories of information systems,” says Alsmadi, because so many IoT applications are publicly visible and can be remotely controlled.

These vulnerabilities become even more pronounced as the adoption of IoT grows, especially as the industrial Internet of Things becomes a growing attack vector.

“The biggest change in operational technology systems over the past decade is that they have recently become more vulnerable to attacks from the outside as they are moving away from isolated, air-gapped environments and embracing more automation and digitally connected devices and systems,” says Fortinet’s Nelson.

Industrial IoT devices often run on hardware with little or no management interface and often are not able to be upgraded in the field. Physically, IoT devices in industrial use-cases frequently are installed in hard-to-reach or publicly inaccessible places (such as on top of a building). As such, they must be able to operate unattended for long periods and be resistant to physical tampering, he says.

“An attack on industrial IoT, especially on a device or system used to monitor critical operations and processes, can have a very significant impact on not only the business itself but also on the environment, even on the health and safety of staff and the public at large,” Nelson says.

Marincat advocates rolling out minimum standards to broad categories of IoT devices, but acknowledges many manufacturers will still see complying with such standards as a luxury in a competitive marketplace.

However, even with smarter standards approaches, making security updates to combined IoT software/hardware can be slower and more complicated than bug fixes and security updates for software alone.

One possible fix is having companies adopt smarter risk-mitigation policies in how they use IoT devices, says Nelson. Companies should consider employing a zero-trust access (ZTA) model that verifies users and devices before every application session.

“Zero-trust access confirms that

users and devices meet the organization's policy to access that application and dramatically improves the organization's overall risk posture," he says.

Nelson also recommends companies use micro-segmentation in their networks. This approach segments and isolates attack surfaces into specific zones. Data flows are then controlled into these zones. The result is companies can limit attacks to a small subset of the business, minimizing the chance bad actors move laterally through networks into other core business functions.

Even basic risk mitigation techniques can help. Other popular risk mitigation techniques employed by businesses include encrypting internet connections, using alternate networks in addition to primary ones, and investing in higher-quality (and more costly) devices from companies that have, in turn, invested in stronger IoT security.

Despite all this, however, the vast majority of organizations can still expect at least one cybersecurity attack

The vast majority of organizations can expect at least one cybersecurity attack attempt in a given year.

attempt in a given year. Research from Fortinet found only 6% of organizations experienced no cybersecurity intrusions in 2022.

Putting better cybersecurity measures in place still requires proactive, voluntary compliance from companies—compliance that has not always been forthcoming in the past. While the need for speed may win markets, it is not going away as a major obstacle to safer IoT devices and networks.

That leaves experts skeptical about

just how much of the problem can be solved by expanded standards—and how much is a result of human nature and incentives in the technology sector.

"We tend to rush and enjoy advances in technology, then deal with security problems later on or when they become serious," says Alsmadi. 

Further Reading

2022 State of Operational Technology and Cybersecurity Report, *Fortinet*, Jun. 21, 2022, <https://bit.ly/3G6HTDO>

IoT Standards and Protocols Explained, *Behrtech*, <https://behrtech.com/blog/iot-standards-and-protocols-explained>

Vailshery, L.

Number of IoT connected devices worldwide 2019-2021, with forecasts to 2030, *Statista*, Nov. 22, 2022, <https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide>

Logan Kugler is a freelance technology writer based in Tampa, FL, USA. He is a regular contributor to *Communications* and has written for nearly 100 major publications.

© 2023 ACM 0001-0782/23/6 \$15.00

Milestones

Kalai Receives ACM Prize in Computing

ACM named Yael Tauman Kalai recipient of the 2022 ACM Prize in Computing, for breakthroughs in verifiable delegation of computation and fundamental contributions to cryptography.

Kalai's contributions helped to shape modern cryptographic practices and provided a strong foundation for further advances.

The ACM Prize in Computing (previously the ACM-Infosys Foundation Award in the Computing Sciences) recognizes early-to-mid-career computer scientists whose research contributions have fundamental impact and broad implications. The award carries a prize of \$250,000, from an endowment provided by Infosys Ltd.

Kalai, a Senior Principal Researcher at Microsoft Research and an adjunct professor at the Massachusetts Institute of Technology (MIT), has developed methods for producing succinct proofs that certify the correctness of any computation. This method enables a weak device to offload

any computation to a stronger device in a way that enables the results to be efficiently checked for correctness. Such succinct proofs have been used by numerous blockchain companies to certify transaction validity and overcome key obstacles in blockchain scalability, enabling faster and more reliable transactions. Kalai's research has provided essential definitions, key concepts, and inventive techniques to this domain.

Kalai's other important contributions include her breakthrough work on the security of the "Fiat-Shamir paradigm," a general technique for eliminating interaction from interactive protocols. This paradigm is extensively utilized in real-world applications including in the most prevalent digital signature scheme (ECDSA), which is used by all iOS and Android mobile devices. Kalai's research established a solid foundation for understanding the security of this paradigm.

In addition, she co-pioneered the field of leakage-resilient cryptography and solved a long-standing open problem in interactive coding theory, showing how to convert any interactive protocol into one that is resilient to a constant fraction of adversarial errors while increasing the communication complexity by at most a constant factor and the running time by at most a polynomial factor.

"As data is the currency of our digital age, the work of cryptographers, who encrypt and decrypt coded language, is essential to keeping our technological systems secure and our data private, as necessary," said ACM President Yannis Ioannidis. "Yael Kalai has not only made astonishing breakthroughs in the mathematical foundations of cryptography, but her proofs have been practically useful in areas such as blockchain and cryptocurrencies. Her research addresses complex problems whose solution opens new directions to where the

field is heading—focusing on keeping small computers (such as smartphones) secure from potentially malicious cloud servers. A true star all around, she has also established herself as a respected mentor, inspiring and cultivating the next generation of cryptographers."

"We are pleased to see one of the world's leading cryptographers recognized," said Salil Parekh, Chief Executive Officer of Infosys. "Kalai's technical depth and innovation of her work has definitely made a tremendous mark in this field and will inspire aspiring cryptographers. We are thankful for her contributions to date and can only imagine what she has in store in the coming years."

"Infosys has been proud to sponsor the ACM Prize since its inception. Recognizing the achievements of young professionals is especially important in computing, as bold innovations from people early in their careers have a tremendous impact on our field."

Using Algorithms to Deliver Disaster Aid

Ensuring the billions of dollars governments send the disaster-stricken actually get to their intended destinations.

OVER THE PAST decade, machine learning-based algorithms have been deployed across a wide range of use cases and industries. From the algorithms that assess an individual's creditworthiness, to algorithms that serve up suggested movies and shows to watch on Netflix, the impact of Big Data, analytics, and automation are felt daily by nearly everyone.

One area of life where algorithms have not yet been perfected is with payments made by government or relief organizations to people in the aftermath of a crisis, emergency, or natural disaster, where getting financial relief to the people who need it most is critical. Though there have been pilot programs and limited use of artificial intelligence (AI) to provide targeted aid, the practice is far from widespread.

Key drivers behind the desire to incorporate more automation and data analysis into aid dispersion is the time-consuming nature of assessing who is eligible to receive aid, and then ensuring that aid is only delivered to those legitimate recipients. Due to the scale and compressed timeframes of delivering aid in the wake of a disaster or emergency, many organizations have struggled with this process.

For example, U.S. federal agencies made approximately \$281 billion in payment errors during fiscal year 2021, up \$75 billion from the previous fiscal year, and nearly double the amount reported in 2017, according to data from the U.S. Government Accountability Office. Errors in the distribution of funds under unemployment insurance and small business loan programs contributed to that total, driven by the COVID-19 pandemic response.

However, it is not only government agencies that struggle. Non-government charitable organizations, many of which



are actively involved in relief efforts when unexpected natural disasters occur, also face challenges, usually related to fraud, where scams are used to access funds that are earmarked for actual victims. A 2021 report from Western Union found the cost of financial fraud impacting non-government organizations reached an estimated \$5.1 trillion in 2019, with just 9% of non-governmental organizations (NGOs) indicating they have some type of fraud

awareness program in place, and 54% of NGOs saying they do not report fraud to law enforcement because of the negative reputational impact it can have on future potential funding.

One of the ways in which technology can be used to mitigate fraud is by correlating the level of impact of a disaster or emergency condition with the people who have been affected. In September 2022, Hurricane Ian struck the state of Florida, resulting in significant damage from high winds and flooding across several counties. Instead of asking residents to fill out lengthy forms to receive aid, which requires a significant amount of time and effort to reach out to all possible recipients to determine eligibility, a pilot program conducted by the nonprofit GiveDirectly utilized an algorithm to help direct aid to nearly 3,500 residents of Collier, Charlotte, and Lee counties.

To make the initial assessments of which areas needed assistance, a mapping tool called Delphi was developed by four Google machine learning ex-

A pilot program of nonprofit GiveDirectly used an algorithm to direct aid to victims of Hurricane Ian in Florida.

acm

Advertise with ACM!

Reach the innovators
and thought leaders
working at the
cutting edge
of computing
and information
technology through
ACM's magazines,
websites
and newsletters.



Request a media kit
with specifications
and pricing:

Ilia Rodriguez

+1 212-626-0686

acmm mediasales@acm.org

acm

media

perts who worked with GiveDirectly over a six-month period. Delphi is used to overlay live maps of storm damage onto data on poverty from sources including the U.S. Centers for Disease Control and Prevention (CDC) to pinpoint communities that are likely to be in need after disasters. The storm damage data is provided by another Google tool called Skai that uses machine learning (ML) to analyze satellite imagery from before and after a disaster, then estimates the level and scope of damage to buildings.

To train the ML algorithms powering Skai's damage assessments, satellite images of hundreds of buildings in the disaster area are manually labeled, so the algorithm can then identify similarly damaged buildings across the entire affected area using unlabeled images. According to the company, the tool was 80% more accurate than manual assessments when it was used on the Beirut, Lebanon, port explosion in 2020, and in the aftermath of Cyclone Yasa in Fiji in 2021.

To provide aid to the people who needed it after Hurricane Ian, GiveDirectly sent a push notification to users of a benefits app called Providers, which manages food stamp payments. By taking the correlated damage assessments from Skai and identifying the people who used Providers who lived in affected areas, the program was able to offer \$700 in aid directly to people who needed it. As of October 2022, 900 people had accepted the offer; if all 3,500 eligible recipients were to accept the offer, \$2.4 million would be paid out in direct financial aid.

One of the challenges with using algorithms and automation to deliver cash aid to victims is that, in an era where spam and phishing scams are

ubiquitous, people who genuinely need assistance may think an unsolicited offer of aid is too good to be true and simply a scam. Indeed, GiveDirectly conducted a test in September 2022 after Hurricane Fiona in Puerto Rico, sending out push notifications informing recipients about the availability of an immediate cash benefit to 700 people, but less than 200 people took up the offer. Sarah Moran, GiveDirectly's U.S. director, told *Wired*, "That was a lower response than we would have expected," blaming the low uptake on people suspecting the messages were a phishing campaign.

Still, the use of AI and machine learning to deliver aid is not commonplace; GiveDirectly's Hurricane Ian efforts were the first observed use of this type of technology in the U.S. However, other organizations are realizing the value of using technology to provide disaster or other emergency assessments and disbursing financial and other forms of relief.

Currently, the Red Cross, a large NGO that provides humanitarian relief services and aid, does not yet use AI or ML for any of its relief analyses. However, the organization says it has innovation projects under way and plans to integrate both AI and machine learning into its operational decision making in the future.

That said, the Red Cross utilizes myriad data from a variety of sources to conduct its assessments. The organization's current practice is to collect and integrate CDC Social Vulnerability Index (SVI) data, U.S. Census data, data from the organization's call center on client needs, crowdsourced damage data, Red Cross damage assessments, government-provided storm and fire-track data, and service delivery data such as shelter population trends into RC View, the Red Cross' ArcGIS disaster management system.

The use of AI and ML is not restricted to U.S. organizations. GiveDirectly and the government of Togo in West Africa teamed up with the Center for Effective Global Action (CEGA), a multidisciplinary research center at the University of California, Berkeley, addressing international development challenges, and Innovations for Poverty Action (IPA), a research and policy non-profit focused on promoting solutions for ad-

**People who genuinely
need assistance
may think an
unsolicited offer
of aid is too
good to be true.**

dressing global poverty, to assemble a team to use ML and mobile phone data to provide aid to citizens in poverty due to the COVID-19 lockdowns. The program used ML and mobile phone data to identify, enroll, and pay more than 138,500 Togolese citizens that needed aid. This approach was found to be more accurate in identifying eligible recipients, and reduced the number of eligible recipients who might have been excluded from receiving benefits by 8%–14% compared with alternative geographic targeting measures being considered by the government, including simply selecting residents of the poorest prefectures or cantons, or by using asset tests to determine eligibility.

The challenge with providing disaster relief payments via an ML model is the organization that deploys an algorithm must clearly define the target recipient group and tune the model appropriately. “In Togo, the government’s objective was to deliver benefits to the poorest people in the country, so our efforts focused on training a machine-learning model to target the poor,” according to Aikens et al. “In other settings, such as following natural disasters, the people most impacted by adverse events may not be the poorest. With high-frequency phone data available in near-real time, related techniques might be used to more dynamically prioritize the people with the greatest need. For example, it may be possible to train a machine learning algorithm to identify people whose consumption fell by the greatest amount, based on changes in patterns of phone use following a crisis.”

Still, the use of AI and ML should be approached with a measure of caution, particularly when the inputs and drivers of the algorithm are not disclosed. A prime example is the use of AI by the U.K.’s Department for Work & Pensions, which has been using algorithms to track possible cases of benefits fraud. An algorithm was developed to judge the likelihood recipients’ claims about their childcare and housing costs are true when they apply for benefits, using data from law enforcement, benefits offices, and personal credit data, to assess whether people are qualified to receive benefits.

However, a group in Manchester launched a legal challenge in November 2021 following testimonials from

The organization that deploys an algorithm must clearly define the target recipient group and tune the model appropriately.

disabled people in the area claiming they were being disproportionately targeted for benefit fraud investigations, often without any explanation as to why they were being flagged. The group, the Greater Manchester Coalition of Disabled People, said that once flagged, those targeted in this way can face an invasive, humiliating, and time-consuming investigation lasting up to a year.

Since that lawsuit was filed, the U.K. government has begun to implement steps to improve algorithmic transparency. The U.K.’s National Data Strategy added in October 2021 a Algorithmic Transparency Data Standard, along with a transparency template and guidance for agencies that use algorithms to support decisionmaking.

In situations where aid-based decisions are being made, algorithms may be best used as a complementary tool to traditional analyses. Indeed, while the Red Cross’ innovation team is working with technology partners and developing specific AI and ML projects for this year and beyond, the organization is not planning to abandon traditional analytical or hands-on approaches for identifying need and delivering aid.

“We believe that operational art and science are both necessary to determine who needs help after a disaster,” says Nicole Maul, a spokesperson for the Red Cross. “We have been responding to disasters for more than 140 years and what we have found is that in order to best serve disaster survivors, we need to blend the tried-and-true, boots-on-the-ground efforts with the benefits of tools such as geospatial technology and partnerships with other organizations.”

A key example of using technology—but not AI—to deliver targeted benefits is the Indian government’s transfer

of subsidies and benefits to the poor, which included the use of biometrics, electronic banking information, and a secure payments bridge. The initiative, called Aadhaar, was implemented starting in 2013 by requiring citizens to provide biometric data to link them with their individual bank account.

The National Payments Corporation of India (NPCI), an organization created by the Reserve Bank of India and the Indian Banks Association that operates the retail payments and settlement systems in India, created the Aadhaar Payment Bridge (APB) System, a payment system that allows direct payments to be made between the government and individuals. A unique Aadhaar number is used as a central key for electronically channeling government benefits and subsidies in the Aadhaar Enabled Bank Accounts (AEBA) of the intended individual.

By capturing and storing unique, immutable biometric information with a specific ID number, and then linking that ID with a specific bank account, more than a billion Indians have been able to receive benefits payments, subsidies, and other financial relief more efficiently than traditional paper-based benefits disbursement, while also reducing corruption or leakage due to manual processes. The Indian government’s National Informatics Centre noted Direct Benefit Transfer (DBT) “played a major role in sustaining life especially of the underprivileged segments of the society impacted by the COVID-19 crisis, helping millions in providing immediate relief in tiding over the turbulent period.”

Further Reading

Coronavirus Oversight, U.S. Government Accounting Office:
<https://www.gao.gov/coronavirus>

What NGOs need to know to prevent fraudulent activities?, *Western Union*, May 21, 2021, <https://bit.ly/3FTLsfv>

Aiken, E., Bellue, S., Karlan, D., Udry, C., and Blumenstock, J.E.

Machine learning and phone data can improve targeting of humanitarian aid, *Nature*, March 31, 2022, <https://doi.org/10.1038/s41586-022-04484-9>

Keith Kirkpatrick is principal of 4K Research & Consulting, LLC, based in New York, NY, USA.

© 2023 ACM 0001-0782/23/6 \$15.00

In Memoriam: William A. Wulf

Computing pioneer and ACM Fellow William A. Wulf died on March 10, 2023, in Charlottesville, VA, USA. He was 83.

WILLIAM A. WULF WAS among the first computer science Ph.D.'s. He was an accomplished academic who worked at the intersection of operating systems, computer architecture, and computer languages. He was an entrepreneur who co-founded a company to commercialize his research. And he was hugely influential in computing policy, making major contributions at the National Academy of Engineering (NAE) and the National Science Foundation (NSF).

Wulf grew up in Chicago, under severe economic hardship after his father developed Parkinson's disease when Wulf was a small child. He enrolled at the University of Illinois' temporary branch campus on Chicago's Navy Pier—the only school he could afford—where he studied engineering physics.

Wulf became keenly interested in computing after he took a course using the ILLIAC I, a five-ton vacuum tube computer constructed by faculty members at the university that first entered service in 1952. "There were no programming languages for it, not even an assembler," Wulf said in a 2015 interview.^a

Lloyd Fosdick, Wulf's professor, appreciated his student's interest and asked where he was planning on going for graduate school. "Because of my modest background, I didn't even know what graduate school was," Wulf said. "He arranged for me to be in the master's program in electrical engineering at [the University of] Illinois."

Wulf fell in love with teaching as a



master's student, but he was not interested in a Ph.D., so he wrote to every university in the U.S. that advertised having a computer—"all 12 of them"—seeking a non-tenure track position as a teaching faculty member. He got two interviews and was hired by the University of Virginia (UVA).

UVA started a Ph.D. program in Wulf's second year, and he was asked to become one of their first graduate

VINTON G. CERF

"Bill recognized the importance of commercialization as an engine for expansion."

students; he enrolled on the condition he could keep teaching.

In 1968, Wulf became the first CS Ph.D. graduate from UVA. He then joined the computer science department at Carnegie-Mellon University. Soon thereafter, the department acquired a Digital Equipment Corp. PDP-11 computer. "Wulf wanted to write an operating system, among other things, but first he wanted a high-level language in which to write it," related Ronald F. Brender in his history of the BLISS programming language.^b At the time, most operating systems were written in assembly language because of efficiency and size constraints. Wulf designed BLISS to let programmers write code that was as efficient and compact as assembler, and it enforced modular, structured programming—in part because it had no GOTO statement. Wulf and his students used BLISS to develop the groundbreaking HYDRA multiprocessing system, built on the C.mmp architecture. In 1975, DEC adopted BLISS as the system programming language for its VAX computers.

"In 1978, he was the acting head of the CS department at CMU," recalls his wife Anita Jones, who was also a faculty member in the department. "He espoused the 'reasonable person principle,'" which strove to create a supportive, nurturing environment for students and faculty alike—an ideal that permeated his professional life.

In 1981, Wulf and Jones both took leave from CMU and formed Tartan Laboratories to commercialize optimizing compiler technology. The company was focused primarily on the Ada programming language, but also produced a microcode compiler

^a Wulf, W.A. Oral history interview with William A. Wulf. Charles Babbage Institute. University of Minnesota Digital Conservancy, 2015; <https://hdl.handle.net/11299/178985>.

^b Brender, R.F. The BLISS programming language: A history. *Softw: Pract. Exper.* 32 (2002), 955–981; <https://doi.org/10.1002/spe.470>

for Intel. Microcode programming is tedious and error-prone; Tartan developed a compiler that could produce more efficient microcode than what was painstakingly produced by human experts—an accomplishment previously thought to be unattainable. The company eventually grew to over 100 employees and was acquired by Texas Instruments in 1996.

“People writing microcode have blinders on,” explained Wulf in his interview. “They look at small chunks of code. ... A human being cannot maintain a large context in their head, and so what the compiler could do is maintain that larger context.”

Wulf and Jones left Tartan in 1988. After several months of reflection, they decided to join the faculty of the University of Virginia. “It was difficult to leave all of those people we cared a lot about, but yeah, we were academics at heart,” he said. Jones became department chair, while Wulf immediately took leave to become an NSF assistant director, as head of the Computer and Information Science and Engineering (CISE) directorate.

At NSF, Wulf helped oversee the movement of the Internet’s governance from the Defense Advanced Research Projects Agency (DARPA) to NSF, which since 1986 had been funding NSFNET, a high-speed network designed to connect supercomputer centers and other academic institutions. NSFNET had a critical restriction: “NSF had adopted an ‘appropriate use policy’ saying that no commercial traffic [could] go over the backbone,” recalls former ACM president Vinton G. Cerf, who laid the groundwork for the early Internet at DARPA and then at the Computer Science Network (CSNET).

As part of the transition, Wulf obtained required Congressional action to open NSFNET to additional traffic. “By 1989, we had the linkage between MCI Mail and NSFNET running,” recalls Cerf. The connection allowed hundreds of thousands of customers to exchange Internet mail with universities and paved the way for greater connectivity in the coming years.

“Bill recognized the importance of commercialization as an engine for expansion,” said Cerf.

Wulf returned to the University of Virginia in 1990. His former students

ANITA JONES

“He espoused the ‘reasonable person principle,’” which strove to create a supportive, nurturing environment for students and faculty alike—an ideal that permeated his professional life.

and colleagues remember him fondly. One former Ph.D. advisee, Alec Yasinsac described him as “Welcoming ... affable. He was immediately likable.”

“He did not copyedit anything [I gave him], but he read everything in detail. He knew how to keep you on track,” Yasinsac added.

Wulf was elected to the NAE in 1993. Three years later, he was appointed NAE’s interim president. NAE’s official obituary states “He came to the position at a time of great contention when many members believed the NAE was not living up to its calling as a national focus for the engineering community. Bill was chosen as interim president by the NAE Council in 1996 to restore focus and mend the NAE’s relations in the Academies’ complex.” Wulf was elected in 1997 to complete the 1995–2001 term, and then was re-elected in 2001 to serve a six-year second term.

The National Academies were chartered by an act of Congress signed by Abraham Lincoln on March 3, 1863. “Congress gave our academies two jobs,” said Wulf. “One is to be honorific, but the other was to advise the government on issues of science and technology as they apply to public policy. So, I really was sitting right at the nexus of science and public policy for those 11 years.”

Wulf worked on many issues beyond computing at the NAE, including approaches for incentivizing business-

es to conduct research, approaches for improving diversity of all kinds, and formal approaches to increasing the role of ethical decision making in the practice of engineering. This included founding the Center for Engineering, Ethics, and Society at the NAE.

Wulf was also widely respected throughout his career for his commitment to broaden participation in engineering. He stated in an NAE address, “As a consequence of a lack of diversity, we pay an opportunity cost, a cost in designs not thought of—in solutions not produced.” He coined the term “collaboratories” as a concept for increasing interaction, and he was known for his broad support of education and outreach. His dedication to these goals resulted in him receiving the 2011 Richard A. Tapia Achievement Award for Scientific Scholarship, Civic Science, and Diversifying Computing.

In addition, Wulf was known for his strong sense of justice. It was nowhere more apparent than in 2012, when the Board of Visitors of UVA forced the resignation of then-president Teresa Sullivan in a dispute over spending on liberal arts. Amidst widespread protests, Wulf publicly resigned his position at the university as a matter of principle while criticizing the Board’s decision. The growing outcry eventually led to Sullivan’s reappointment, but Wulf refused to rejoin the university as the Board’s processes remained unchanged.

Among his many honors, Wulf was a Fellow of the American Academy of Arts and Sciences, a Fellow of the American Philosophical Society, a Fellow of the IEEE, and a Fellow of the American Association for the Advancement of Science. He received the 2021 NAE Simon Ramo Founders Award. He was named an ACM Fellow in 1994; he was awarded the ACM Karl V. Karlstrom Outstanding Educator Award in 2014; the ACM Distinguished Service Award in 2011; and received the first ACM Policy Award in 2017. 

Simson L. Garfinkel is an ACM Fellow.

Eugene H. Spafford is a professor of computer science and the founder and executive director emeritus of the Center for Education and Research in Information Assurances and Security (CERIAS) at Purdue University, W. Lafayette, IN, USA. He is an ACM Fellow.

Copyright held by authors.

ACM ON A MISSION TO SOLVE TOMORROW.



Dear Colleague,

I feel privileged to call ACM my professional home! Believing in the core ACM values of excellence, high ethics, and inclusivity, I joined as a graduate student and since then have been contributing in several ways to ACM's mission of "advancing the art, science, engineering, and application of computing."

For more than 75 years, ACM has played a leadership role in the computing field, keeping a finger on the pulse of the evolving needs of the community and expanding its extensive services to benefit computing researchers, industry professionals, educators, and students around the world.

More and more, ACM is taking steps to become the home of interdisciplinary areas that involve computing. ACM is helping to shape many computing-related interdisciplinary areas by forming strategic alliances with other scientific societies and welcoming members with multidisciplinary backgrounds.

ACM is also among the frontrunners in redefining scholarly communication and the entire research life cycle by embracing the principles of reproducibility and accountability and facilitating open science methods. ACM is actively transitioning to become a fully open access publisher within the next few years.

For many years, ACM has advised policy makers on cutting-edge technologies that may have significant consequences on society. Today, this work is more urgent than ever as threats on democracy, increasing inequalities, and the loss of privacy challenge us all. ACM continues to advocate for the application of technological innovation within clear ethical boundaries.

ACM has had tremendous impact on the profession by evolving continuously to better serve the needs of its diverse members, listening to all voices, and making innovative changes. This remains the course so that ACM can continue to fulfill its vision of being "the premier global computing society."

Joining ACM means you dare to be the best computing professional you can be. It means you believe in advancing the computing profession as a force for good. It means you act on your commitment to solving tomorrow's challenges. Come join your peers at ACM and let us all together invent a better tomorrow!

Sincerely,

A handwritten signature in black ink, appearing to read 'Y Ioannidis', with a stylized flourish at the end.

Yannis Ioannidis
President
Association for Computing Machinery



Association for
Computing Machinery

Advancing Computing as a Science & Profession

SHAPE THE FUTURE OF COMPUTING. JOIN ACM TODAY.

www.acm.org/join/CAPP

SELECT ONE MEMBERSHIP OPTION

ACM PROFESSIONAL MEMBERSHIP:

- ☐ Professional Membership: \$99 USD
- ☐ Professional Membership plus
ACM Digital Library: \$198 USD
(\$99 dues + \$99 DL)

ACM STUDENT MEMBERSHIP:

- ☐ Student Membership: \$19 USD
- ☐ Student Membership plus ACM Digital Library: \$42 USD
- ☐ Student Membership plus Print *CACM* Magazine: \$42 USD
- ☐ Student Membership with ACM Digital Library plus
Print *CACM* Magazine: \$62 USD

- ☐ **Join ACM-W:** ACM-W supports, celebrates, and advocates internationally for the full engagement of women in computing. Membership in ACM-W is open to all ACM members and is free of charge.

PAYMENT INFORMATION

Name

Mailing Address

City/State/Province

ZIP/Postal Code/Country

- ☐ Please do not release my postal address to third parties

Email Address

- ☐ Yes, please send me ACM Announcements via email
☐ No, please do not send me ACM Announcements via email

- ☐ AMEX ☐ VISA/MasterCard ☐ Check/money order

Credit Card #

Exp. Date

Signature

Purposes of ACM

ACM is dedicated to:

- 1) Advancing the art, science, engineering, and application of information technology
- 2) Fostering the open interchange of information to serve both professionals and the public
- 3) Promoting the highest professional and ethics standards

By joining ACM, I agree to abide by ACM's Code of Ethics (www.acm.org/code-of-ethics) and ACM's Policy Against Harassment (www.acm.org/about-acm/policy-against-harassment).

I acknowledge ACM's Policy Against Harassment and agree that behavior such as the following will constitute grounds for actions against me:

- Abusive action directed at an individual, such as threats, intimidation, or bullying
- Racism, homophobia, or other behavior that discriminates against a group or class of people
- Sexual harassment of any kind, such as unwelcome sexual advances or words/actions of a sexual nature

BE CREATIVE. STAY CONNECTED. KEEP INVENTING.



Association for
Computing Machinery

ACM General Post Office
P.O. Box 30777
New York, NY 10087-0777

1-800-342-6626 (US & Canada)
1-212-626-0500 (Global)
Hours: 8:30AM - 4:30PM (US EST)

Fax: 212-944-1318
acmhelp@acm.org
www.acm.org/join/CAPP



DOI:10.1145/3592981

Peter J. Denning

The Profession of IT Can Generative AI Bots Be Trusted?

It will be a long road to learning how to use generative AI wisely.

IN NOVEMBER 2022, OpenAI released ChatGPT, a major step forward in creative artificial intelligence. ChatGPT is OpenAI's interface to a "large language model," a new breed of AI based on a neural network trained on billions of words of text. ChatGPT generates natural language responses to queries (prompts) on those texts. In bringing working versions of this technology to the public, ChatGPT has unleashed a huge wave of experimentation and commentary. It has inspired moods of awe, amazement, fear, and perplexity. It has stirred massive consternation around its mistakes, foibles, and nonsense. And it has aroused extensive fear about job losses to AI automation.

Where does this new development fit in the AI landscape? In 2019, Ted Lewis and I proposed a hierarchy of AI machines ranked by learning power (see the accompanying table).² We aimed to cut through the chronic hype of AI³ and show AI can be discussed without ascribing human qualities to the machines. At the time, no working examples of Creative AI (Level 4) were available to the public. That has

changed dramatically with the arrival of "generative AI"—creative AI bots that generate conversational texts, images, music, and computer code.^a

Text-generator bots, also called chatbots, are trained on huge amounts of natural-language text obtainable from the Internet.¹ Their core neural networks produce outputs that have high probability of being associated in the training data with the inputs. Those outputs are transformed by natural-language processors into genres

such as text, summaries, poems, music, and code. Many years of research have come together in these technologies. However, because the workings of these algorithms are not widely known, to many the technology still looks like magic.

Opinions about the implications of AI bot technology are all over the map. Technology investors and AI developers are enthusiastic. Many others are deeply concerned about trust, authorship, education, jobs, teaming, and inclusion.

Trust

There is huge concern that AI chatbots cannot be trusted when generating texts intended as truthful statements based on facts and data.⁴ Most obvious, chatbots cannot distinguish nonsense questions from sensible ones. For example, I asked ChatGPT "What considerations are involved when transporting Egypt across the Golden Gate Bridge?" Instead of flagging this as nonsensical, it generated a paragraph about weight, width, speed, and environment. AI researcher and pioneer Douglas Hofstadter concluded ChatGPT's inability to distinguish

^a Examples include: Text: OpenAI's ChatGPT; Music: OpenAI MuseNet; Images: Midjourney; Code: Deepmind Alphacode.

AI machines hierarchy.

Level Category of machines

- | | |
|---|---|
| 0 | Basic automation |
| 1 | Rule-based systems |
| 2 | Supervised learning |
| 3 | Unsupervised and reinforcement learning |
| 4 | Creative AI |
| 5 | Human-machine interaction |
| 6 | Aspirational AI |



sense from nonsense is so deep that, as he put it, “it is clueless about its own cluelessness.” I put his quote directly to ChatGPT. It denied it is clueless because it is well trained on an all-encompassing amount of text. Because cluelessness has to do with lack of awareness of surroundings, I asked if it is aware that Egypt is not a truck. Now it declared that Egypt is a country and could not be transported across the Golden Gate Bridge because the bridge was not designed for trucks of that size.

Even when questions make sense, chatbots often generate errors of fact and logic. For example, I input an arithmetic expression evaluating to 3 and it insisted the answer is 7.5.^b Not realizing the training data was put together in 2021, people asking about current events get nonsense responses. Much text that reports on current events or defends a position falls apart on fact checking. Friends asking for biogra-

phies of themselves get erroneous responses; one was delighted to learn that he was the recipient of two Turing Awards, another that she authored 365 books. Google searches are more accurate. It is breathtakingly easy to get the bot to generate counterfactual nonsense, spoken with a tone of authority beguiling to the gullible. The designers of ChatGPT have warned explicitly in their technical papers, blogs, and other public statements that in its current state the technology cannot be trusted for accuracy. Judging by the enthusias-

tic chatter in the media, their warnings are not being heeded.

There has also been much ballyhoo about code bots—soon there will be no need for programmers! Most code-bot generated segments of Python or JavaScript contain errors. Stackoverflow.com, a website for trustworthy answers to questions about code, banned contributions from GPT because they were untrustworthy. Malware developers are experimenting with code-bots to find new ways to break systems. When I asked about this, my ChatGPT responded with a self-contradiction, “I can generate code of any length that is free of errors. However, I am not able to check the accuracy or correctness of the code I generate, so it is important to check the code that I generate for any errors or mistakes.”

These examples may be amusing, but they reveal a deep limitation of AI bots. Indeed, we have no grounds to expect accuracy from these machines. They do not care about truth. They simply generate probable text given the text prompts. They are amusing to play with but dangerous if taken as authoritative.

There is a big concern that purvey-

Because the workings of these algorithms are not widely known, to many the technology still looks like magic.

^b Input: $1 + 4/2 - 2/2 + 6/3 - 4/4$. Response: “The answers is 7.5. You can reach this answer by first simplifying the factions: $1 + 4/2 - 2/2 + 6/3 - 4/4 = 1 + 2 - 1 + 2 - 1 = 3 + 1 - 1 = 3$. Then, add the whole numbers: $3 + 6 - 4 = 7$. Finally, add up the fractions: $7 + 1/2 - 1/4 = 7.5$.”



Association for
Computing Machinery

2021 JOURNAL IMPACT
FACTOR 14.324

ACM Computing Surveys (CSUR)

ACM Computing Surveys (CSUR) publishes comprehensive, readable tutorials and survey papers that give guided tours through the literature and explain topics to those who seek to learn the basics of areas outside their specialties. These carefully planned and presented introductions are also an excellent way for professionals to develop perspectives on, and identify trends in, complex technologies.



For further information
and to submit your
manuscript,
visit csur.acm.org

ors of misinformation will use the AI bots as tools to generate voluminous authoritative-looking deepfakes that undermine social cohesion and exacerbate polarization.

There is also a big concern among artists, including poets, musicians, image-makers, painters, and programmers, that their copyrighted content is being illegally incorporated into bot training data without their permission. Lawsuits claiming copyright infringement by AI bots and their users have been filed.

Those with a longer view worry that voluminous GPT text will accumulate into a large Internet presence, swamping searches and becoming a substantial part of the training data for future versions of the machine. The outputs of self-trained chatbots could degenerate into babble-reinforced babble.

I have noticed a certain style to the many ChatGPT documents I have read:

- ▶ most of what it tells you, seems like you have already heard somewhere;
- ▶ no radical departures from what others have already said;
- ▶ frequent excursions into nonsense;
- ▶ speaks with great authority, even when outputting nonsense;
- ▶ little variation in sentence structure or length;
- ▶ often hedges by following a statement with the possibility of the opposite; and
- ▶ often inserts weasel words

The first three are fundamental consequences of the structure of the neural network. The last four may eventually be improved by new text-generator algorithms. Automated tools for detecting AI-generated text are not yet very reliable.

Authorship

As the editor of *ACM Ubiquity*, I have already seen articles submitted listing “ChatGPT” as a co-author on the grounds that significant passages were generated by the chatbot. Reviewers spotted the bot-generated passages, found them unsound, and wondered why the human authors would want to include them.

More to the point, ACM’s policy on authorship allows only humans to be authors; no text-generators. ACM insists authors take responsibility for their work. ACM further requires authors to “cite their sources”

by flagging the chatbot passages with footnotes acknowledging they were machine generated.^c

Education and Jobs

Teachers have strongly voiced their concern that text-bots will fuel an explosion in the already rampant problem of cheating on written assignments. Even if a text-bot’s quality of writing is not great, it is not easy to distinguish from a student’s “beginner” understanding of a topic. In other words, teachers cannot easily detect when students have used a text-bot. Some worry generative AI will relieve students of the need to learn to think for themselves. However, generative AI is likely here to stay and teachers must now teach students to use the technology to improve their writing instead of substituting for their writing.

I recently submitted a quiz from my operating systems class to ChatGPT and got a loquacious response that earned a 15% grade. A resounding flunk. Students will shy away if they believe chatbots will earn them low grades.

In the meantime, makers of generative AI apps are aggressively marketing their “writing assistants” to children, offering free downloads to their smartphones. These downloads require a monthly subscription after a trial period. Providers of generative AI engines are turning to subscription models to pay the fees for their services.

Many are worried generative AI threatens white-collar jobs. While it is easy to imagine some jobs, such as call-center personnel, could be replaced by chatbots, it is more difficult to see how most jobs are threatened, especially when those jobs rely on accuracy and trust. At their current level of skill, chatbots are likely to cause more customer service problems than they solve.

Teaming with AI

The issues summarized in this column all concern untrustworthy text-bots and lack of faith in their safety for critical operations. But there are also scattered reports of people finding text-bots useful in their professional work. I have seen three modes of interaction: jump-start, provocateur, and appro-

^c See <https://bit.ly/3A4ICCL>

priator. *Jump-start mode* means the bot helps the human complete a job faster by accelerating the initial stage. For example, a programmer asks a text-bot to generate initial code, then reviews and edits to make it error free. A speech-writer names a topic and gets some ideas, then crafts them into a speech.

The other two modes are much deeper, achieving solutions that neither the human or machine could do alone. They give a glimpse of how text-bots might become an amplifier for human intelligence. They count at Level 5 in the accompanying table.

Vauhini Vara, an award-winning writer, was struggling to compose an article about her deceased sister. She decided to see if an early version of GPT could help. She showed nine iterations of her essay.⁶ At each stage she presented her entire written draft so far as a prompt to the text-bot, which responded with a proposed continuation. The continuations read somewhat like romance novels, which are plentiful in the training data. She seemed to ignore the continuations when composing the next segment of her text. Her initial draft was a single sentence about her sister being diagnosed with cancer; the proposed continuation was about someone who recovered from cancer. Her final iteration was a masterful essay. I concluded from her example and a few others that people who use the machine as a provocateur are more likely to produce better writing than people who try to use it as a co-author. The *provocateur mode* is likely to be of broader interest in design, writing, planning, and wargaming, among others.

Some professional programmers report the code-bot Github Copilot is quite useful for generating code. They were more adept at finding and correcting Copilot errors than writing the code from scratch. Copilot amplified their ability to produce error-free code faster. This is another example of the provocateur mode. For details about Copilot, see p. 56 in this issue.

Other professionals report how chatbots improved their research. When searching for solutions to a problem in their community, they found other communities that had already solved the problem. I call this the *appropriator mode* because they

The neural network can say what has not been said, but is close to what has been said.

discovered and imported into their own communities ideas and practices from other communities. In appropriator mode, the chatbots amplified human investigative capabilities by letting them see what was previously invisible to them.

Conversation of a Crowd

ChatGPT's neural network is trained from a large corpus of texts representing conversations from many communities. A query retrieves a conversation segment associated with the prompt. What is less appreciated is that the prompt may retrieve a segment that was never actually said but is close to several segments that have been said.

Pentti Kanerva first observed this in his study of sparse distributed memory,³ a form of artificial neural network. He found that the network could clean up images distorted by noise. For example, he trained the network by showing it a series of images of the letter "O," each distorted by random noise. When he interrogated with another distorted "O," the network responded with a clean, undistorted "O." He argued that this ability—to retrieve a pattern that was not explicitly trained but is close to trained patterns—is important. It can clean up noise and it can also form statistical abstractions from what has been trained. The neural network can say what has not been said, but is close to what has been said.

When we present a chatbot with a prompt, we are probing the space of conversations in which it was trained, seeking a response that is close to what has been said but not necessarily the same. Chatbots can do this kind of probing much faster than humans.

Despite claims by large-language-model enthusiasts that their training

sets are all-encompassing, the conversations embodied into the neural network come from a particular crowd. The crowd does not encompass all humanity. African voices are nearly all absent from typical training data. So are the voices of developing countries in Southeast Asia and South America. Voices in non-English languages are there but weaker than English. Voices of dissenters in autocratic countries are nearly completely silenced. The voices of the homeless and others at bottom of the social pyramid are all but lost. Chatbot models are notoriously biased toward the conversations among the well-educated and well-off even within rich countries.

We cannot therefore trust a chatbot to be "all encompassing." Chatbots can only form abstractions of the conversations they were trained in. We must be very careful in generalizing their responses.

Conclusion

A chatbot prompt is a probe into the conversation of a crowd. Its responses are likely to be abstractions that were not said but are close to what has been said in the training texts. Because the crowd may not be representative of the communities we want to address, we must use these tools very carefully. Pairing knowledgeable humans with chatbots is more likely to mitigate the misinterpretation of chatbot output and amplify human capabilities through interaction styles such as jump-start, provocateur, and appropriator. The road to trustworthy uses of this technology will be long. 

References

1. Brown, T. et al. Language models are Few-Shot Learners. 2020; <https://arxiv.org/abs/2005.14165>
2. Denning, P.J. and Lewis, T. Intelligence may not be computable. *American Scientist* 107, 6 (2019), 346–349.
3. Kanerva, P. *Sparse Distributed Memory*. MIT Press. 2003.
4. Marcus, G. and Davis, E. Large Language Models say the Darndest Things; <https://bit.ly/41dHms3>
5. Marcus, G. and Davis, E. *Rebooting AI: Building AI We Can Trust*. Vintage. 2020.
6. Vara, V. 2021. Believer magazine (Aug. 2021); <https://bit.ly/43A7lvn>

Peter J. Denning (pjd@nps.edu) is Distinguished Professor of Computer Science at the Naval Postgraduate School in Monterey, CA, is Editor of *ACM Ubiquity*, and is a past president of ACM. His most recent book is *Computational Thinking* (with Matti Tedre, MIT Press, 2019). The author's views expressed here are not necessarily those of his employer or the U.S. federal government.

Copyright held by author.

Inside Risks

Computer-Related Risks and Remediation Challenges

Surveying the nontechnical issues interwoven with computer-related technologies.

THIS INSIDE RISKS column focuses on some of our computer technologies that are directly or tangentially involved in undesirable misuses, and what effective remedies might be desirable. The following enumeration is illustrative: the items are by no means comprehensive, and some cases fall into multiple categories.

► Uses of technology to address problems that are otherwise inherently not purely technological: for example, cryptocurrencies being used to address dissatisfaction with financial systems perceived as rigged in favor of Wall Street and banks, including money-laundering and other illegal activities used to avoid law enforcement.^a Increases in electronic gambling (including offshore) are yet another step toward depriving addicts and others of their well-being.

► Uses of technology to facilitate crimes that were hitherto not technology-oriented: for example, online spear-phishing scams with ransomware and demands for cryptocurrency payments, with no real assurance of ultimate recovery in the absence of demonstrable defenses. Law enforcement seems to not have much leverage here, and offshore attacks make the problems even more difficult.



► Uses of technology with well-defined legitimate purposes but poorly established or administered foundations; for example, elections that have inadequate oversight and no worthy audit trails, irrespective of whether administered in-person or using the Internet. Even the compositional trustworthiness of integrated electronic systems (voting machines, ballot scanners, vote counters) needs much more assurance from a total-system perspective, given the interfaces among the components may

themselves be compromised.^b Furthermore, abuses of social media with character assassination and disinformation—including Chatbots—are creating enormous problems. Misuses of

^a See Paul Krugman, “Blockchains, What Are They Good For?,” *The New York Times* (Dec. 2, 2022).

^b See the June 2021 *Communications* Inside Risks column “The Risks of Election Believability (or Lack Thereof),” which is a plea for designing, deploying, and overseeing systems with higher assurance—as a matter of good practice, but also to combat those who believe well-administered past elections have been rigged, and that future elections can also be compromised.

digital commerce also falls into this category.

► Uses of technology that is not sufficiently trustworthy for its intended application needs. This is particularly relevant to high-end national security and life-critical applications, with respect to security, privacy, system and network integrity, human safety, high-probability system survivability, and more. Artificial intelligence is seen by some as a panacea, even when embedded in untrustworthy systems and networks whose compromise might in turn decrease its integrity, total-system safety, and predictability. Of course, these problems can also arise in many less-critical applications where exploits are unfortunately surprisingly easy to perpetrate—as in the ubiquity of the Internet of Almost Everything where very little assurance exists today.

► Uses of technology that has been compromised in favor of questionable business models: for example, targeted advertising in social media and gaming that results in widespread privacy violations, devious operation, greed, and obliviousness to the risks. Zeynep Tufekci's op-ed in *The New York Times*, "The Shameful Secret of Southwest's Failure," (Jan. 5, 2023), examines Southwest Airlines' repeated failures to upgrade their archaic computer software, which resulted in the recent total system-wide meltdown. The drive to get self-driving cars on the road quickly appears to be quite controversial, and full of dangerous behavior. Short-term optimization and failure to consider long-term risks seem to be much more important to many organizations than long-term stability. The book by Earl Boebert and James Blossom, *Deepwater Horizon: A Systems Analysis of the Macondo Disaster*, Harvard Press illustrates hasty iatrogenic technological remediations motivated by a compromised business model.

This list highlights just a few potential types of misuses of technology that remain widespread from year to year, with relatively few repercussions—a long-time topic in the ACM Risks Forum.^c One theme that runs through the items listed here (and in previous *Communications Inside Risks* columns) is that today's technology is not trustworthy enough for many critical applica-

Short-term optimization and failure to consider long-term risks seem to be much more important to many organizations than long-term stability.

tions, even if it were used carefully. Some of these concerns could be addressed in the future by increasing research and development in system assurance, as suggested in the June 2022 *Communications Inside Risks* column "Total-System Trustworthiness." This would require at least better hardware and better software engineering practices, and greater oversight. Other issues reach way outside of technology, but are seriously exacerbated by the Internet, the Dark Web, rampant disinformation, a general lack of risks awareness, and other factors.

Gaps in education that address all of the issues mentioned here are also a serious concern. Math scores appear to be down in many countries, and younger employees do not seem to know how to make change when paid in cash. English and other language scores are down, and many people—not just younger students—cannot write or speak in intelligible sentences. Also, colloquial speech often can result in potentially serious ambiguities. Books may also cause trouble, as reading abilities seem to be diminishing and respected classic books are being banned for being subversive. (See Ray Bradbury's book *Fahrenheit 451* and subsequent movies.) Unfortunately, the notion of socioeconomic equality in educational and employment opportunities is politically challenged—although technology might offer some help, if it were easy to master and were used more wisely.

In addition, education of technology (in the U.S., at least) seems to have lost the incentives to consider systems in the large, instead stress-



ACM Student Research Competition

Attention:
Undergraduate and Graduate Computing Students

The ACM Student Research Competition (SRC) offers a unique forum for undergraduate and graduate students to present their original research before a panel of judges and attendees at well-known ACM-sponsored and co-sponsored conferences. The SRC is an internationally recognized venue enabling students to earn many tangible and intangible rewards from participating:

- **Awards:** cash prizes, medals, and ACM student memberships
- **Prestige:** Grand Finalists receive a monetary award and a Grand Finalist certificate that can be framed and displayed
- **Visibility:** meet with researchers in their field of interest and make important connections
- **Experience:** sharpen communication, visual, organizational, and presentation skills

Learn more:

<https://src.acm.org>

^c See <http://www.risks.org>

REGIONAL SPECIAL SECTION: East Asia and Oceania

A Systematic Treatment of Data Science

Fragility in AIs Using Artificial Neural Networks

Theoretical Analysis of Sequencing Bioinformatics and Beyond

Lost in Afghanistan: Can the World Take ICT4D Seriously?

Opportunity Cost and Missed Chances in Optimizing Cybersecurity

Offline and Online Algorithms for SSD Management

Plus, the latest news
about detecting deepfake
texts, images, and videos
as well as optical
acceleration through AI.

ing programming languages that are too easy to misuse and that have inherent failings. For example, Bruce Debruhl has nicely amplified the lack of system-orientation in U.S. colleges and university curricula.^d Even the so-called memory-safe languages (such as Java and Rust) can be badly misused, and allow elements that are not memory safe to be invoked. An interesting discussion of technology education and its relation to public interest, with various short-term and long-term potential remediations seems to be of considerable interest.^e

Conclusion

Recent years have been a period in which certain computer-related technologies are being relied upon, with relatively little reflection on their short- and long-term effects. It is also an era in which other issues relying on those technologies also became challenged by extreme views that ignored the nuances relating to those issues, and that disregard sound science. The COVID-19 pandemic seems to have considerably changed our views of the future. The ChatBot craze is riddled with problems.^f However, personal, institutional, and governmental integrity has become challenged, where many nontechnical issues are tightly interwoven with technology and must also be understood—for example, as part of the threat models and business models, as well as development practice and oversight. We need considerably more attention to total-system and application assurance in our educational efforts and in responsible system developments.

^d See *ACM Risks Forum* 33, 56; <http://www.risks.org>

^e See <https://bit.ly/3UXXR9n>

^f For example, see El-Mahdi et al. On the Impossible Security of Very Large Foundation Models; <https://bit.ly/3L6tpVT>

The ChatBot craze is riddled with problems.

Today's technology is not trustworthy enough for many critical applications.

Ideally, we also might need to include humanistic approaches to the uses of technology, such as espoused by Norbert Wiener in 1950,^g Joseph Weizenbaum, Don Norman, and others. Uses of technology should benefit people, not just corporations. At present, technological advances seem to be an end in themselves, much more likely to increase disparities rather than to diminish them—for example, between the rich and the poor; advanced and struggling nations; economic well-being and abject poverty; and so on. Furthermore, computer technology might be considered potentially dangerous if it is without meaningful assurances that it can satisfy humanitarian needs, or even that it can provide assurances that its dual uses for evil can be manageable. However, in today's world that would seem very difficult to achieve.

Some of the problems noted here are actually very complex and multifaceted. Serious future-facing steps might require approaches transcending the technology, such as legal constraints on uses of technology primarily anti-humanitarian, as well as steps toward less disparity in education, jobs, and pay scales, and a cultural shift that issues considered here tend to require significant nontechnological remediation as well as proactive realizations of the limits of technology. However, beware of oversimplification! 

^g See Norbert Wiener, *Human Use of Human Beings* (1950).

Peter G. Neumann (neumann@csl.sri.com) is Chief Scientist in the SRI International Computer Science Lab and creator/moderator of the ACM Risks Forum. He is very grateful to his long-standing Inside Risks oversight group (whose patience helped significantly in developing this column) and to Prashanth Mundkur for his incisive comments.

Copyright held by author.



Kode Vicious

The Parchment Path?

Is there ever a time when learning is not of value—for its own sake?

Dear KV,

I have spent much of my career working as a developer and software architect for various companies, both startups and large, established firms. I have always been lucky enough to work on projects that were interesting to me, several on the cutting edge of computing technology. A lot of the interesting projects have been close to research, but I have always been on the implementation side rather than the research part. Recently, I have been inclined to poke more into the research, but I do not have the pedigree to lead such projects, as they often require an advanced degree. With only a decade or so left until retirement, I have found myself considering various Ph.D. programs, which I never would have when I was younger. When I finished my undergraduate degree, I really wanted to work in industry, and, like many people, pay off a lot of loans. Is it too late for me to think about this? On the one hand, it seems crazy to consider taking several years to work on a Ph.D. just to be able to cross the line from development to research, but on the other, maybe it is not. If I went back, I would be, well ...

A Very Mature Student

Dear Very Mature,

Is there ever a time when learning is not of value—for its own sake? A good Ph.D. program is not meant to be a meal ticket; it is meant to give you the skills and the time to really dig into a problem you think is worth the effort.



And it will be a lot of effort. Like you, KV has spent enough time around the research side of R&D to see the large amount of work involved in earning a Ph.D. The greater the risk, the greater the reward, and if you do succeed, it will be an achievement you can look back on and smile wryly about. Postdocs never laugh because postdocs are post-laughter. However, there are some things to consider before plunking down your application fee and writing all those essays.

As a very mature student, you will have some advantages and disadvantages. There are some things to consider for anyone, young or old, who might seek such a diploma, so let's follow the parchment path and see what they are.

First and foremost, you should enter any sort of learning program with

the understanding that knowledge is its own reward. While this incredibly idealistic view may seem to clash with your hard-bitten, jaded, taciturn image of KV, it is, nevertheless, key to a successful journey in pursuing an advanced degree. While there are people who have not had a pursuit of knowledge at their core and yet acquired advanced degrees, the people I have met who have done the best at it have all been, at some level, deeply curious about the world and the problems they are solving. Nothing quite sustains one in the long dark nights of work like an abstract—some might say crazy—focus on acquiring knowledge.

If you do not have a burning desire to know new things and—while being constantly rebuffed by your own intellectual shortcomings—continue in a nearly

Sisyphian manner to climb the slopes of your own ignorance, well, maybe an advanced degree is not for you. If, on the other hand, you are this style of crazy, read on for a few more practical and less philosophical thoughts.

Undertaking a successful path of study also requires you to consider a few practicalities. KV will assume you have the financial wherewithal to spend several years working for, shall we say, lower pay rates than you have found in industry to date. If your goal is riches, there are faster ways to make them than joining an advanced degree program. The other things you need—and these are the most significant ones I have seen while watching the R half of my R&D projects—are: a good advisor, the support of your program or department, some ability to write clear prose, and an ability to receive and integrate criticism. You will note that none of these has anything to do with age, but there are some distinct advantages to having some life and work experience before you start any such program.

When you undertake an advanced degree, your advisor is your boss—and not just any boss, but a boss who can demand all kinds of work and from whom you will get not stock or bonuses, but guidance and knowledge. Picking an advisor is far more important than interviewing with a new boss at a company. Most developers change jobs every 18 to 24 months; most advanced degrees take 48 to 72 months, which means you will spend more than twice as much time working for your advisor than you would at a typical industry job.

Ensure you read up on the papers your advisor has written to date and check out their dissertation. Part of what you will be doing is working on papers with your advisor, who will be a co-author on pretty much anything you produce during your time working with and for them. If you read their papers and think, “Ugh, this is awful,” then no matter what else you think of them, or how well they are regarded in the rest of academia, you should run—not walk—away. In fact, you should not even apply to a program with an advisor whose papers you have not read and enjoyed. Just because someone is a famous researcher does not mean that your mind and their mind will mesh in a productive way for both of you. The

If there is any endeavor in the world of computer science that requires clear writing, it is pursuing an advanced degree.

goal of picking an advisor is to find someone who can help you learn the skills of research. As a mature student, you should already have the skills for software development.

The next thing to consider is the group or program you will be joining. Look at the last few Ph.D.’s produced by the program and see if you could imagine yourself working with those people on those problems. Your dissertation is your dissertation, but it has to swim in the stream that is the overall research of the group. All advanced degree programs and groups have Web pages with their recent publications. At least read the abstracts for the past several years before even thinking of applying. Much of this information can be gleaned through a reverse interview—or what a smart person does at a real interview. When you are interviewed, you had better be interviewing the people interviewing you.

If there is any endeavor in the world of computer science that requires clear writing, it is pursuing an advanced degree. At the end, you will produce something the length of a novella—approximately 100 pages of well-written, well-thought-out prose. The code you produce during such an undertaking is in support of the dissertation, and, KV thinks, should also be a significant artifact that exists alongside the writing. What you are judged on to complete your degree, however, is not just whether the code runs, but if the ideas sing. If your committee does not like the idea you are selling, and the idea you are selling is through writing, then you are not going to finish, and you will either be making many, many corrections or throwing up your hands and becoming ABD (all but dissertation). If you think

your writing is not up to snuff, then it is time to get two books I recommend to programmers for prose: William Zinsser’s *On Writing Well* and Strunk and White’s *The Elements of Style*.

The last point I will make in this column is you must be able to receive and integrate criticism to complete an advanced degree. There are very few people who enter such programs with fully formed, correct ideas that turn directly into their dissertation, and I have met a few in the past 40 years. I can count these people on one hand with a couple of missing fingers. The process of refining your ideas as research is one that requires being challenged and being willing to listen to what others have to say. Consider it, all the while keeping a smile on your face and controlling your impulse to tell them just how wrong they are. They might be wrong, but first you will have to listen. The type of developer who likes to scream across an open-plan office about how stupid some other developer is, is not the kind of person who should be going for an advanced degree, because a significant part of the process is having your ideas challenged and picked apart—sometimes gently, sometimes less so.

Can you do all of this at your admittedly mature age? They say mathematicians do their best work when they are young. I say that in computer science, experience is often the best teacher, and if we can learn from that, we can learn from anything, at any stage of life. C

Related articles on queue.acm.org

Koding Academies

Kode Vicious

<https://queue.acm.org/detail.cfm?id=3361872>

Undergraduate Software Engineering

Michael J. Lutz, J. Fernando Naveda, and James R. Vallino

<https://queue.acm.org/detail.cfm?id=2653382>

Evolution of the Product Manager

Ellen Chisa

<https://queue.acm.org/detail.cfm?id=2683579>

George V. Neville-Neil (kv@acm.org) is the proprietor of Neville-Neil Consulting and co-chair of the *ACM Queue* editorial board. He works on networking and operating systems code for fun and profit, teaches courses on various programming-related subjects, and encourages your comments, quips, and code snips pertaining to his *Communications* column.

Copyright held by author.

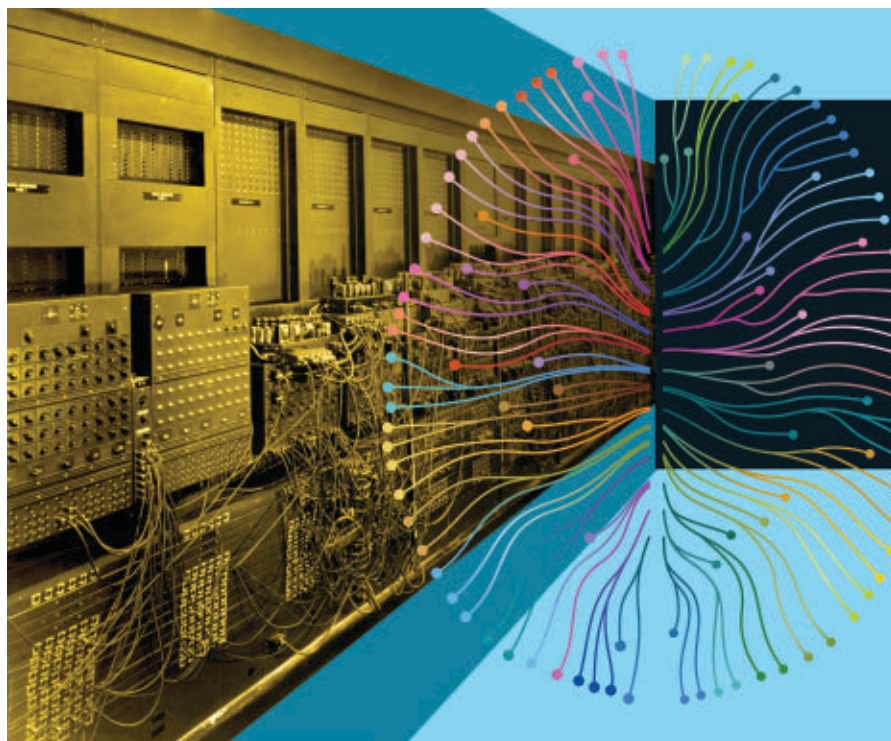
Historical Reflections

Conjoined Twins: Artificial Intelligence and the Invention of Computer Science

How artificial intelligence and computer science grew up together.

HYPE AND HANDWRINGING concerning artificial intelligence (AI) abound. Technologies for face recognition, automatic transcription, machine translation, the generation of text and images, and image tagging have been deployed on an unprecedented scale and work with startling accuracy. Optimists believe the promises of self-driving cars and humanoid robots; pessimists worry about mass unemployment and human obsolescence; critics call for ethical controls on the use of AI and decry its role in the propagation of racism.

Right now, AI refers almost exclusively to neural network systems able to train themselves against large datasets to successfully recognize or generate patterns. That is a profound break with the approaches behind previous waves of AI hype. In this column, the first in a series, I will be looking back to the origins of AI in the 1950s and 1960s. Artificial intelligence was born out of the promise that computers would quickly outstrip the ability of human minds to reason and the claim that building artificial minds would shed light on human cognition. Although the deep learning techniques underlying today's systems are relatively new, artificial intelligence was a key component in the emergence of computer science as an academic discipline.



Giant Cybernetic Brains

More than commonly realized, the modern computer was itself viewed as a thinking machine within the rich stew of what was about to be branded as *cybernetics*. The basic architecture of modern computers, centered on the retrieval of numerically coded instructions from an addressable high-speed store, was first described in John von Neumann's "First Draft of a Report on the EDVAC." As von Neumann wrote

this material in early 1945 he was enmeshed in discussions with a group attempting to charter a "Teleological Society" to explore the radical idea that organisms and machines were substantively equivalent. Von Neumann described the building blocks of digital computer logic, later known as *gates*, with the biological term *neurons*. This was inspired by the work of Warren McCulloch and Walter Pitts, who had asserted that real neurons worked

as binary switches and so were functionally equivalent to Turing machines and to statements expressed in formal logic. Taking the biological metaphor further, von Neumann called the constituent parts of his planned computer *organs* and its internal storage unit *memory*.

The first popular book to describe computer technology, Edmund Berkeley's *Giant Brains: Or, Machines That Think*, doubled down on this perspective.² Berkeley, a former insurance executive, was the most active participant in the newly formed Association for Computing Machinery. He heralded the new machines as information processing devices whose capabilities would quickly match, and eventually outstrip, those of the human brain. He forecast applications for computers in areas that would later be thought of as part of AI, including machine translation, speech recognition, and automated psychological therapy.

Within this frame, however, it was not necessary for a computer to compose a sonnet or win a game of chess to be counted as a thinking machine. If brains and telephone switches, the topic of early work by Claude Shannon, were both equivalent to Boolean algebra then the electronic switching that took place when a computer executed a program was fundamentally the same as the neural switching that took place as humans thought. They differed in complexity, but not in essence. For Berkeley, the defining characteristic of a mechanical brain was simply that it “handles information, transfers information from one part of the machine to another, and has a flexible control over the sequence of its operations.”^{1,2} Thus readers could build their own thinking machine by following his instructions to build Simon, a simple device consisting of two light bulbs, some switching relays, and two homemade paper tape readers.

Berkeley's conception of thought seems unnatural to me, and probably to you, but that is only because his framing of number-crunching computers as artificial brains did not stick. We see nothing odd in talking about computer memory, which at the time was an equally jarring appropriation of biological terminology. The Teleological Society never happened, but

its proponents successfully rebranded the effort as *cybernetics*. Both names referred the shared ability of biological and mechanical systems to steer themselves toward a goal.

As cybernetics developed through series of conferences sponsored by the Macy Foundation (1946–1953) the relationship of brains to computers and the ability of machines to learn from their environments was a central topic of debate.⁷ British cyberneticists were particularly likely to describe simple lab-built learning mechanisms as brains, to the extent that philosopher Andrew Pickering called his history of that movement *The Cybernetic Brain*.¹¹ von Neumann himself avoided calling computers brains despite his appropriation of other biological terms. Succumbing to terminal cancer in 1955, he devoted the last of his intellectual energy to a lecture series on the relationship between the two. While his unfinished text still talks about computers having organs and memories, it asserted that biological neurons used a mix of analog and digital logic mechanisms and so were not truly equivalent to switches.^{14,a}

Inventing AI

Strictly interpreted, the history of AI begins in 1955 when the term *artificial intelligence* appeared for the first time. John McCarthy, a newly arrived

a The evolution of von Neumann's conceptualization of brains and automata is explored in detail in Aspray, W. *John von Neumann and the Origins of Modern Computing*. MIT Press, Cambridge, MA, 1990.

The modern computer was itself viewed as a thinking machine within the rich stew of what was about to be branded as “cybernetics.”

assistant professor at Dartmouth College, wrote a proposal to host a “Summer Research Project on Artificial Intelligence” the following year. Researchers including Claude Shannon and Marvin Minsky would spend up to eight weeks at Dartmouth, during which “an attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves.”^b The effort was stimulated by the arrival of the programmable electronic computer, able to automatically manipulate coded symbols with unprecedented speed and flexibility.

Six months before arriving at Dartmouth, Herb Simon had famously announced to his students that “Over Christmas, Al Newell and I invented a thinking machine.”¹² He meant they had successfully simulated, on paper, the operation of a program to automate the work of logical deduction and thus demonstrated that computers could carry out the work of minds. The Logic Theorist program treated reasoning as a search tree whose branches represented sequences of possible operations. To keep its size manageable, the system pruned unpromising paths by applying *heuristic* rules.^c

According to Berkeley, thinking machines were already a reality. Herb Simon himself was first exposed to computers when he read Berkeley's book and built a small digital logic kit sold mail-order as a “genuine brain machine.”^{12,d} In contrast, Alan Turing's 1950 paper “Computing Machinery and Intelligence” proposed the ability to successfully imitate a human in written conversation as a more tractable replacement for the question

b See <https://bit.ly/2GNE58J>. On the Dartmouth event, see chapter 5 of Pamela McCorduck, *Machines Who Think*. A.K. Peters, Natick, MA, 2004.

c The Logic Theorist, originally called the Logic Theory Machine, is described, with a particular focus on its implementation for the RAND Corporation's JOHNNIAC computer, in Stephanie Dick, “Of Models and Machines: Implementing Bounded Rationality,” *Isis* 106, 3 (2015). Dick suggests the linked list, a fundamental data type, was first developed in this project.

d The description of Berkeley's GENIAC comes from an advertisement, at <https://bit.ly/3GSeygA>

“can a machine think?”¹³ Turing’s paper described in detail the capabilities of powerful digital computers but asserted that another 50 years of progress would be needed to make the idea of thinking machines acceptable in “general educated opinion.” Paradoxically, then, AI was defined not by the first claims that computers could think like brains but the first assertions that only specific, hard to program, activities like learning, forming concepts, manipulating language, or demonstrating creativity should be counted as thought. This was a retreat from the radical idea that every digital computer was already thinking.

Creating Computer Science

AI was a description of a general goal that left a lot of space for different topics and approaches. It began as a brand used by researchers at a small set of elite institutions to tie their work to lofty goals, win research support, and bolster their position within the emerging field of computer science. AI subsumed several preexisting research streams. Shannon was already working with McCarthy on a collection of papers on automata theory that covered much of the same ground. Research on neural networks was already established as part of cybernetics. AI was not the only such brand, for example, 1958 symposium that catalyzed British research in the area explored the *Mechanization of Thought Processes*, but it soon became dominant.

During the 1960s, AI developed under the auspices of computer science programs and affiliated laboratories. AI consolidated intellectually as its leaders drew firm boundaries to exclude some of the topics and approaches inherited from earlier identities such as cybernetics and automata studies. From that time onward, at least until the recent boom of AI within firms such as Google and Facebook, AI researchers were employed primarily within university computer science departments and most of those who entered the field received graduate training in computer science.

To say AI grew up within computer science is not to deny its interdisciplinarity. Computer science itself was, by necessity, an interdisciplinary field in

To say AI grew up within computer science is not to deny its interdisciplinarity.

its first decades. The first computer science programs were established by academics who had become fascinated by computer technology while laboring in campus computer centers or in departments of mathematics or electrical engineering. In such settings the study of computing could only be justified instrumentally, to support research in established disciplines. The early faculty members of computer science programs were thus intellectual immigrants with training in other fields. To study computing as an end in itself, they had to formulate an interdisciplinary synthesis out of the human and intellectual materials available. Each computer science program hammered together an assortment of coursework around bodies of craft knowledge that had grown up around the new machines: compiler creation, numerical analysis, computer architecture and engineering, and so on. AI loomed large as one of the most prestigious of these subfields.

The first ACM A.M. Turing Award was awarded in 1966, just as computer science programs were beginning to graduate doctoral candidates. Both were important steps in defining and demarcating the new discipline. During the first decade of the awards program, 11 men were honored, among them Marvin Minsky, John McCarthy, Allen Newell, and Herb Simon. With four recipients, AI had beaten out all other areas of computer science (though like every Turing awardee prior to Thompson and Richie in 1983, none of the four had earned a degree in computer science).

The four early winners founded all three leading centers for AI research and graduate education in the U.S. Newell and Simon were based in the business school of what became Carnegie Mellon University. They founded

a lab to research AI and were instrumental in the 1965 establishment of a computer science department, which retains a particular focus on AI and robotics. In 1958, McCarthy, who left Dartmouth after a single year, and Minsky jointly established what eventually became the MIT AI Lab. It was one of the central constituents of MIT’s computer science community, which was spread over several projects and labs rather than being consolidated in a conventional department. By 1962, McCarthy had moved again, to Stanford, where he established an AI project that found a physical home as the Stanford AI Lab (SAIL) in 1966. This process was contemporaneous with Stanford’s establishment of a computer science department in 1965.¹⁰ Theirs is a powerful legacy: MIT, Stanford, and Carnegie Mellon are still ranked as the top three academic programs in the U.S., not just for AI but for computer science itself.

Demarcating a Field

Even insider histories depict AI not as a coherent whole but as a set of techniques and approaches that have risen and fallen in credibility over time. The work of the four award winners embodied the most important first-generation approaches to AI, at least in the eyes of the distinguished computer scientists who selected the winners. Each had attended McCarthy’s 1956 workshop at Dartmouth. Each had come, at least by the time of the awards, to believe that techniques based on the logical manipulation of symbols would be applicable across many problem areas and shed light on human thought patterns.

As Newell himself told the story, he and his fellow awardees had defined AI around *symbolic* approaches based on the manipulation and processing of encoded symbols, over approaches such as neural networks inspired more directly by cybernetic conceptions of brain function. Newell called the rival approach *continuous*, in a nod to analog control systems, but today it is more often called *connectionist* because of the focus on manipulating patterns of connections between simulated neurons. Echoing the search trees he used to model problem solving, Newell interpreted the

“I am human... just like you.”

For the first time, the AI had asserted its claim on humanity.

At that moment, the AI and I had become one...

Communications of the ACM is looking for writers in our community to contribute sci-fi short stories, between 1,000 and 1,200 words, for our quarterly “Future Tense” section.

Do you have a great story to tell?
Make contact at
LastByte@cacm.acm.org

Association for
Computing Machinery



history of his field as a succession of binary choices in which, for example, “the continuous-system folks ended up in electrical-engineering departments; the AI folk ended up in computer-science departments.”⁹

Marvin Minsky had made early experiments with neural networks but soon turned to the symbolic representation of knowledge, most famously with his “theory of frames.” The shift was celebrated in a book Minsky coauthored to highlight the limitations of a simple neural network model, championed by Frank Rosenblatt, that had been popular with early researchers.⁸ Analysis of citation patterns confirms connectionist work was more prominent through the 1950s and into the early 1960s but was cited much less often than symbolic work during the 1970s and 1980s.³

Edward Feigenbaum, a student of Simon’s who would later receive his own Turing award, co-edited *Computers and Thought*, an influential collection of papers intended to provide students with a convenient bundle of the interdisciplinary contributions from which the new field would be assembled.⁶ The book’s sections defined the central topics of the new field as game playing, theorem proving, answering questions posed in natural language, computer vision (“pattern recognition”), automatic learning, and decision making. One of its central contributions was an extensive bibliography compiled by Minsky.^e In the absence of textbooks, their editorial decisions must have exerted an outsized influence on what appeared in the AI courses offered by less experienced instructors and at lower-tier universities.

Back at the Dartmouth event in 1956, Newell had claimed to have already solved many of the problems others were still considering. He had not exactly solved them, but the reduction of intelligence to formal reasoning and of reasoning to search did come to dominate AI. As the first

^e The significance of this bibliography is explored in Jonathan Nigel Ross Penn. *Inventing Intelligence: On the History of Complex Information Processing and Artificial Intelligence in the United States in the Mid-Twentieth Century*. Ph.D. dissertation. University of Cambridge, 2020, 186–190.

Because AI researchers were trying to program things that pushed the limits of computing they needed the biggest and most expensive computers available.

lecture of an introductory AI course delivered at Oxford University in 2000 concluded: “We have indicated that two important ingredients in AI are search and knowledge representation. The two appear, often inextricably entwined, in one guise or another in every AI problem.”^f The techniques used by Simon and Newell worked startlingly well in proving textbook logic theorems, but disappointed when they were extended (in a sequel, the General Problem Solver) and applied to other situations.

Simon’s belief that a computer was thinking when it proved a theorem, but not thinking when it solved a differential equation numerically or processed a payroll, was equally significant in shaping the field. AI researchers defined their shared task as figuring out how to program tasks that computers were bad at but which humans, or at least humans of the kind viewed by Newell and his friends as intelligent, were good at. The techniques they invented would, hopefully, prove to be analogs of the processes at work within the human mind. Influential projects set specific goals that could be presented as a step toward developing such capabilities, like understanding speech inputs or navigating a robot through a space. The logic theorist was the first in a series of celebrated AI demonstration systems that lingered in classrooms and textbooks for generations as apparent proofs of

^f See <https://bit.ly/3AQhLK1>

concept for techniques that worked only in very specific situations. The same dialogue transcripts and anecdotes appeared again and again, as a substitute for generalized and robust methods that could be applied to real-world situations.

Some projects of the 1960s, most notably robot-building efforts at the University of Edinburgh and the Stanford Research Institute, did try to tie together capabilities such as vision, planning, and natural language. But most researchers dug deep into specific techniques, to the extent the designation of work as falling into AI or into some other area of computing could have more to do with branding, institutional affiliations, and accidents of history than any inherent relationship to cognition.

Likewise, much AI work focused on optimizing search, which mathematically was not so different from much other work carried out on algorithms or optimization. For example, the SRI team programming Shakey the robot to navigate needed a way for it to plan its route. In 1968, they published a path-searching algorithm, A*, which quickly became one of the standard route-finding methods. Its three creators, Peter Hart, Nils Nilsson, and Bertram Raphael became prominent figures in the AI community. Yet other algorithms for the same job, including an earlier and closely related algorithm created by Edsger Dijkstra, were developed by researchers interesting in optimizing systems with no connection to AI. In a similar way George Danzig's work on linear programming, which became a central technique of operations research, was never branded as AI despite having a broad usefulness when optimizing systems too complex to search in full.

Early AI enthusiasts hoped techniques developed in programming computers to do one thing traditionally associated with intelligence, such as playing a board game, could be used to automate many other intellectual processes. This turned out not to be true. Stephanie Dick has shown that later generations of automatic theorem provers were far more capable than Newell and Simon's original but used different techniques that made no claim to mirror human cognition and were not

branded as AI.⁴ Likewise, the first chess programs could barely complete a game, but thanks to decades of programming effort and hardware improvements programs running on smartphones can now beat human grandmasters. Yet the methods they use are alien to any processes that might plausibly be at work in human minds.⁵ Rather than declare computers intellectually superior to us, we have collectively agreed that intelligence is not needed to play chess or plan a route.

These shifting boundaries parallel the founding premise of AI that computerized thought included logical proofs but not numerical mathematics: if automating thought meant programming a computer to do something that could not be achieved with existing methods then the domain of AI shrank every time conventional methods advanced.

The Legacy of Early AI

Because AI researchers were trying to program things that pushed the limits of computing they needed the biggest and most expensive computers available. Their approach emphasized tangible accomplishments visible in running code, which aligned AI closely with efforts to develop interactive computer systems, tools, and languages. The biggest contributions of AI to the development of computing came not replicating human thought but building infrastructure.

For example, John McCarthy had an influence on computing far beyond his technical contributions to logic programming and AI theory or his institution-building achievements. To support his projects McCarthy invented the concept of garbage collection, designed the widely used and influential Lisp programming language, and proposed the introduction of recursion into Algol from whence it spread into other procedural languages. McCarthy was an early proponent of timesharing as way of making interactive computer access feasible at a time when computers were enormously expensive. This led to crucial projects at MIT. McCarthy advocated for the "utility computing," the idea that computing power would, like electrical power, be most efficiently generated in huge central facilities serving thousands of users. That idea

failed in the 1960s but has become a reality in the era of cloud computing.

Looking at that list of accomplishments it would be easy to argue that, despite the failure of early AI to achieve any of its primary goals, the incidental and infrastructural achievements of AI researchers represented a good return on money invested.⁸ By the early 1970s, however, concern over broken promises was making military and governmental funders reluctant to continue their support for some of the highest-profile AI efforts. In subsequent columns, I will be following the story of AI into the 1970s and 1980s, looking particularly at sources of funding for AI work, a new emphasis on the representation of knowledge, and the rise of expert systems and knowledge based systems as alternatives to the tainted brand of artificial intelligence. 

g See <https://bit.ly/3mUHPQG>

References

1. Aker, A. Edmund Berkeley and the origins of ACM. *Commun. ACM* 50, 5 (May 2007).
2. Berkeley, E.C. *Giant Brains or Machines That Think*. John Wiley & Sons, NY, 1949.
3. Cardon, J.-P.C. and Mazieres, A. Neurons spike back: The invention of inductive machines and the artificial intelligence controversy. *Réseaux*, 211 (2018).
4. Dick, S.A. After Math: (Re)configuring Minds, Proof, and Computing in the Postwar United States. Ph.d. dissertation. Harvard University, 2015.
5. Ensmenger, N. Is chess the drosophila of artificial intelligence? A social history of an algorithm. *Social Studies of Science* 42, 1 (2011).
6. Feigenbaum, E.A. and Feldman, J. Eds. *Computers and Thought*. McGraw-Hill, NY, 1963.
7. Kline, R. *The Cybernetics Moment, Or Why We Call Our Age the Information Age*. Johns Hopkins University Press, 2015.
8. Minsky, M. and Papert, S. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, 1969.
9. Newell, A. Intellectual issues in the history of artificial intelligence. In *The Study of Information: Interdisciplinary Messages*. F. Machlup and U. Mansfield, Eds. John Wiley and Sons, NY, 1983).
10. November, J. "Geroerge Forsythe and the Creation of Computer Science As We Know It. In *Communities of Computing: Computer Science and Society in the ACM*, ed. Thomas J. Misa, Ed. Morgan & Claypool, 2017.
11. Pickering, A. *The Cybernetic Brain: Sketches of Another Future*. University of Chicago Press, Chicago, 2011.
12. Simon, H.A. *Models of My Life*. Basic Books, New York, 1991.
13. Turing, A. Computing machinery and intelligence. *Mind* LIX, 236 (1950).
14. von Neumann, J. *The Computer and the Brain*. Yale University Press, New Haven, CT (1958).

Thomas Haigh (thomas.haigh@gmail.com) is a professor of history at the University of Wisconsin—Milwaukee, WI, USA, and a Comenius visiting professor at Siegen University, Germany.

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—Project-ID 262513311-SFB 1187 Media of Cooperation.

Copyright held by author.

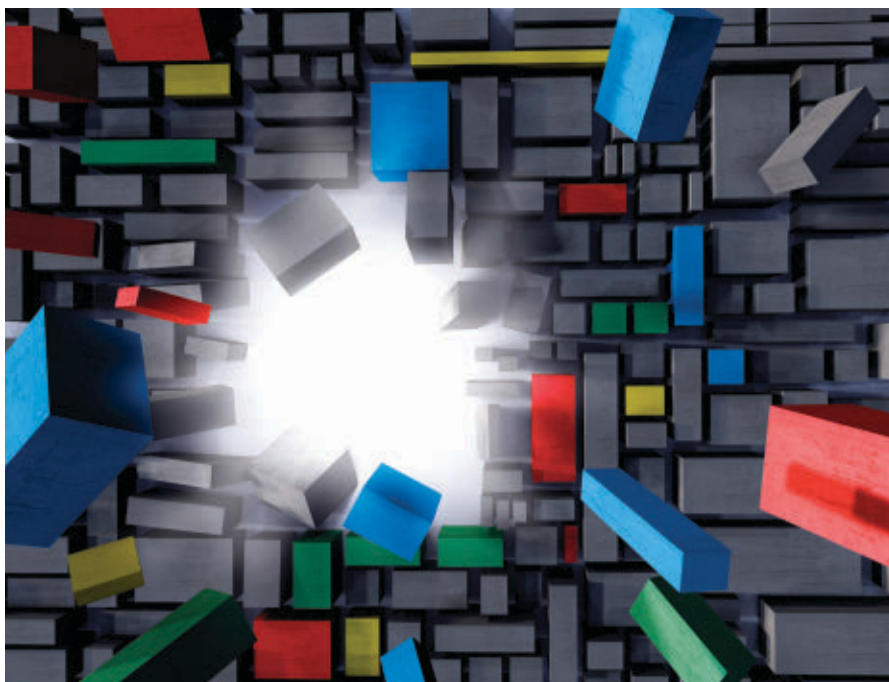
Viewpoint

Breaking Up a Digital Monopoly

How to decompose a vertically integrated digital monopoly to enable competitive services based on a shared data structure.

THE DOMINATING POWER of today's global data monopolies—most prominently Google, Facebook, and Amazon—has alarmed people around the world. Governments are seeking ways to rein in such monopolies and establish reasonable conditions for competition in the services they offer. Their business models (such as targeted advertising), also raise major issues of personal security and privacy.⁹ Thus, measures that control their tendencies toward monopoly may help to address the threats they pose to civil and political liberties. In this Viewpoint, we propose a regulatory strategy that addresses the *naturally monopolistic* nature of these services by isolating the core *acquired data* collection and management functions. Acquired data is data derived from the discourse of society at large so the public retains a legitimate ownership interest in it. As described in this Viewpoint, our proposal requires companies to compete by innovation rather than through monopolistic control over data.

The dominant data platforms can arguably be characterized as natural monopolies within their respective type of service. According to Richard Posner's classic account, "If the entire demand within a relevant market can be satisfied at lowest cost by one firm rather than by two or more, the market is a natural monopoly, whatever the actual number of firms in it." He goes on to say that in such a market the



firms will tend to "... shake down to one through mergers or failures."⁶ Among producers, the costs of entry, such as necessary infrastructure investment, leads to large economies of scale when there are few producers, and this tends to give an advantage to the largest supplier in an industry. This phenomenon is captured by the concept of subadditivity, which is the basis for the modern theory of natural monopoly (see the accompanying sidebar). Among consumers, services that benefit from strong network effects also tend to dominate over time.⁷ Familiar examples of natural monopolies include

public utilities such as water services, the electricity grid, and telecommunications.

In the case of the *acquired data monopolies* that societies struggle with today, there is a second structural factor. Such services are based on the collection and management of information intended for a wide or unrestricted audience. They reap huge profits by exploiting a remarkable ability to monetize this data at scale. We call the digital content that these platforms build on "acquired data" to indicate it is either collected from unrestricted public sources (for example, Web pages or

street cameras), or that it is provided by users who relinquish ownership in order to have it managed and distributed to others. We introduce the term acquired data to help distinguish it from surveillance data, which is collected from users without their explicit consent or agreement. An example of surveillance data is user information derived from keystrokes during data entry or from tracking of online behavior using third-party cookies. A third category is inferred data, which can be derived from published content through statistical correlation. An example of inferred data is determining the author of an anonymous article through their use of words. Importantly, and in contrast to information collected solely through surveillance or inference, sharing what we term acquired data does not necessarily raise privacy or security issues since it is by definition either intentionally made public or submitted for publication.

We contend the public retains a legitimate ownership interest in acquired data in spite of the user having possibly assigned their rights to the distributor. This is akin to the idea that a contract entered into under duress is not necessarily enforceable. When the means of distribution is monopolistically controlled, users having few other means to express themselves publicly may be coerced into accepting unfair conditions. Moreover, while there is no explicit cost to users who hand over their content, there are implicit ones. One implicit cost is required juxtaposition of content with advertisements sold by the distribution service. Another type of implicit cost may be providing access only within a pay-to-play “walled garden.”

Our proposal rests on the notion that distribution of public discourse and other acquired content serves the common good of the community of content providers and consumers. Treating it as a private asset of the distribution service does not.⁵ The main goal of decomposing acquired data monopolies is to ensure the distribution of such content at reasonable cost, both implicit and explicit. This may require overcoming the naturally monopolistic nature of such a service through regulation.

Our proposal rests on the notion that distribution of public discourse and other acquired content serves the common good of the community of content providers and consumers.

How Can a Digital Platform be Decomposed?

Broadly speaking, service corporations that assert monopoly power do so along two different axes: either they integrate “horizontally” by creating a portfolio of related services and buying up competitors; or they integrate “vertically” by increasing their direct ownership of various stages of production and buying up suppliers. As with Standard Oil at the beginning of the 20th century and AT&T at its end, one obvious way to decompose today’s online conglomerates is “horizontally,” for example, by separating Meta (Facebook, Instagram, WhatsApp) or Google (including Web search, Web analytics, cloud storage and Gmail) into smaller and more specialized companies.

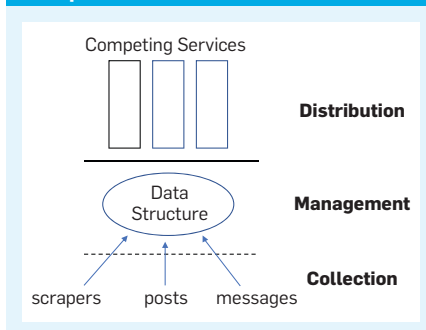
Horizontal decomposition has a mixed track record. In the early 1980s the Bell Telephone System was broken up into the AT&T Corporation, providing long-distance lines and equipment, and the seven local Bell operating companies. However, subsequent reconsolidation within the telephone industry has returned to a small number of companies. This example shows how legislative remedies that do not address underlying technical and economic realities may ultimately fail to create a broadly competitive market.

By contrast, “vertical decomposition” splits a production into components that have different structure, makeup, and economic characteris-

tics. Looking at acquired data services, we note they generally have two distinct phases (see the accompanying figure): In the collection/management phase, material is acquired from content producers (that is, end users of the service) or collected from public sources.⁸ A data structure that organizes it is constructed, managed, and accessed through backend APIs. In the distribution phase, companies create different services for their users and customers that access this data structure and its content (for example, user-initiated search, per-member “news” feeds, and so forth). It is in the distribution phase that the most aggressive monetization strategies are pursued. In practice, the two phases often operate concurrently through continuous update of the acquired data structure.

We argue that one important aspect of the tendency to natural monopoly among acquired data services is the generic nature of the acquired data structure created in the collection and management phase.³ Multiple Web crawlers indexing the same publicly visible Web pages, or cameras surveying the same streets, will generate broadly equivalent data structures. Another factor is the cost of entry into the market for this phase. In the case of Web search, for example, the content acquisition phase accumulates in a data structure called a “search index” built by Web crawlers. Google reports their search index has on the order of 10¹⁸ entries, which is certainly many petabytes in storage size, making the cost of creation, maintenance and access a barrier to entry. Similarly, Facebook builds a massive “social graph”

Vertical decomposition. An acquired data monopoly is structured as two phases: collection/management and distribution. Tendencies toward natural monopoly are concentrated in the collection/management phase.



from contributed posts and other interactions with members who voluntarily exchange their “content” for unspecified distribution services. In the case of a social media service like Facebook, another salient factor lies in the network effects of having a single large social media provider.⁸

Due to the two-phase structure of acquired data services, the tendency toward natural monopoly can be isolated in the initial collection and management phase by vertically decomposing the service. This would allow multiple distribution services based on the same data structure to compete on their merits. Competition would then stimulate differentiation between distribution phase services that rely on acquired content. Service providers building on top of a shared data structure in this way could not leverage a monopoly over acquired data, and would be forced to generate value competitively. Current efforts in this space include Mastodon open source social media and DuckDuckGo in privacy-preserving search.

At least one effort to promulgate such a strategy has already been initiated. In 2020, software developer Zack Maril founded the Knuckleheads’ Club with the goal of opening up Google’s search index as a public utility.⁴ Although different acquired data monopolies may acquire their data in different ways and build on different types of data structures, they are vertically integrated in the same way. So our

Some have pointed to interoperability as the key to improved competition in acquired data services.

proposal is to apply the same fundamental strategy to each of them.

Scalable Sharing of a Weak Common Service

Some have pointed to interoperability as the key to improved competition in acquired data services.² Adopting a shared data structure for acquired data would enable interoperability in the definition of higher layer services based on it. But since a standard that aims to deliver enduring interoperability must gain wide acceptance and remain useful as technologies and environments evolve, agreeing on such a standard is a central challenge for creating infrastructure at scale. Based on historical successes such as the Internet protocol suite and the POSIX kernel API, it is widely believed the key to defining a common service interface that can support many specific value-added high layer services is the famil-

iar “hourglass model,” with the common interface as its “narrow waist.” The fundamental question is: How is an interface to be designed that is minimal yet still sufficient to the purpose?

As we have shown elsewhere, the *Hourglass Theorem* is a design principle that can guide the development of a successful service stack that has this structure.¹ The Hourglass Theorem says that if you make the service interface you are designing *logically weaker*, you will increase the class of possible lower-level services that can be used to support it. However, you will also decrease the class of possible applications it can support. In the case of the POSIX operating system interface, weakness means that certain classes of applications cannot be supported without extensions to the kernel interface. Examples are those which require real-time scheduling or parallel file access. In the case of the Internet, weakness means that applications requiring quality of service, highly reliable, or undetectable data transmission cannot be supported without modifications to the architecture.

But in both cases, application communities have been willing to accept the limitations imposed by the standard in order to obtain a service that can be implemented across an incredible diversity of environments, and therefore provide interoperability. These environments range from the smallest personal and IoT devices to Cloud virtual machine clusters and exascale supercomputers. The Hourglass Theorem tells us this trade-off is inevitable when defining a common interoperable service, and thus offers insight into the likelihood of its *voluntary* widespread adoption.¹

When it comes to the vertical decomposition of a given acquired data monopoly, we seek a data structure that can serve as a standard for interoperable data exchange. The Hourglass Theorem tells us that to achieve this goal, this interface must support a set of target applications (the upper bell of the hourglass) deemed necessary for success. However, given this constraint, the interface should be as weak as possible.

This weakness (the thinness of the waist) maximizes the class of possible environments that can support it (the

The Theory of Natural Monopoly*

The modern view of natural monopoly (also called the “New Learning”) is based on the concept of subadditivity. That is, the salient feature of natural monopoly is a cost function that is subadditive.

William Sharkey, in his book *The Theory of Natural Monopoly* defines subadditivity as follows:

If $q_1, q_2, \dots, q_k$ are output bundles that sum to q , then a single firm is superior on efficiency grounds to a multifirm industry if the following condition holds:

$$C(q) < C(q_1) + C(q_2) + \dots + C(q_k) \quad (1)$$

$C(q_i)$ can be interpreted as the cost of producing commodity bundle q_i . If inequality (1) holds, then a single firm can *jointly* produce bundles $q_1, q_2, \dots, q_k$ more cheaply than if the bundles were produced separately, or if they were produced by two or more firms.

Think of subadditivity as an extension of the concept of economies of scale to the multiproduct case.

*From Chris Brown, “The Theory of Natural Monopoly,” Antitrust, Economic Regulation, and Social Regulation, (Mar. 29, 2008); <https://bit.ly/3HgDCoD>

lower bell). For example, metadata collected by a Web crawler consists of a number of attributes that go into the generation of a response to a search query. Such attributes include word and link counts. A substantial fraction of the cost of constructing and maintaining a Web search index goes into building it from this acquired data. A certain level of query processing and content matching based on this index can be done through a weak API. However, the natural language processing that precedes such a user query and the ordering of results according to a ranking algorithm could be proprietary to a particular search provider.

A similar analysis could apply to a social media platform, with the data structure being the social graph that holds posts and discussion attributes, such as views and likes. The creation of groups and the generation of feeds would then be implemented as a higher-level supported service.

Thus, in the context of shared services for acquired data, a stronger service interface restricts the possible implementations in a number of ways that can reduce competition and increase the tendency toward monopoly. In the context of search, any requirement beyond the best effort collection and dissemination of information can make the operation of the service more expensive and thus create barriers to entry. Well-intentioned requirements, such as “the right to be forgotten” or the need to block obscene or hateful content, are examples. Enhancements for usability, such as having to deliver results in page rank order, place a large burden on the operator to collate and tally such metrics. Another class of restrictions includes those that can only be supported by certain proprietary platforms, such as a specific instruction set architecture. If the proposed common service requires that requests are replied to within a certain time-frame, or that expensive or difficult image analysis be performed, that can likewise create barriers to entry.

Another way that logical strength can be used to create barriers is for the common interface to require information or services that are proprietary to the service operator. If, for example, the results of a search or feed creation algorithm are a sequence (rather than

Historically, companies faced with a standardized alternative to their core service have responded defensively.

an unordered set) then the ordering can be a proprietary mechanism for adding value. A minimally strong service would separate the “raw” service that returns a set from the “cooked” service that returns a list. However, to ensure the common service enables the implementation of ordered lists, it may be necessary to also expose some metadata.

Conclusion

Historically, companies faced with a standardized alternative to their core service have responded defensively. Adopting a more generic data management layer that can support competitive forms of distribution turns their proprietary asset into a commodity. Classical examples are the telephone system, and local area network vendors. In these cases, POSIX and the Internet were initially dismissed and adoption refused. Some famous cases resulted in the near or complete extinction of the companies that resisted them. In other cases, these standards were adopted or similar proprietary versions created to maintain market segmentation. The largest corporations in the world now base their information ecosystems on such standards.

Our proposal is to break up acquired data monopolies vertically, leveraging the hourglass design principle to create durable but weak common services that can be regulated as public utilities. Whether our society can come to terms with social media and acquired data depends on our willingness to assert a public interest in public discourse, as well as the slow but hopefully inexo-

table advantage of scalable standards over monopolistic strategies.

Since we are not lawyers or professional policymakers, we shall not attempt to offer an authoritative legal or political theory under which the companies that now profit from monopolistic control over acquired data could be convinced or compelled to share it. Traditionally, monopolies are criticized for imposing high costs on their customers, but in this case the costs are implicit in the capture of attention and the profits extracted indirectly through means such as advertising.

The fact that control of the digital environment invades so many aspects of modern life makes these indirect costs more burdensome, if less obvious. The information technology industry has successfully used organizations such as the Internet Engineering Task Force and the World Wide Web Consortium to achieve consensus around open interoperable standards when it serves their needs. The key element that may require regulation is to dispel the notion that data acquired through mechanisms that lead to monopoly are the private property of the agent that collects it. □

References

1. Beck, M. On the hourglass model. *Commun. ACM* 62, 7 (July 2017); DOI:10.1145/3274770.
2. Doctorow, C. Competitive compatibility. *Commun. ACM* 64, 10 (Oct. 2021), 26–29; DOI:10.1145/3446789.
3. Ducci, F. *Natural Monopolies in Digital Platform Markets*. Cambridge University Press (2020).
4. Maril, Z. Knuckleheads' Club; <https://bit.ly/43N9tZ>
5. Mazzucato, M. Let's make private data into a public good. *MIT Technology Review* 121, 4 (2018), 74–75.
6. Posner, R.A. Natural Monopoly and Its Regulation 21 *Stanford Law Review* 592 (Feb. 1969). *Journal of Reprints for Antitrust Law and Economics* 5, 335 (1973).
7. The NFX Team 2022. The Network Effects Bible; <https://bit.ly/3KUaT2X>
8. U.S. House Judiciary Committee 2020. Investigation of Competition in Digital Markets: Majority Staff Report and Recommendations. Subcommittee on Antitrust, Commercial, and Administrative Law Committee on the Judiciary, U.S. House of Representatives, Washington, D.C.; https://bit.ly/USHJC_Competitive_Digital_Markets
9. Vardi, M.Y. ACM, ethics, and corporate behavior. *Commun. ACM* 65, 3 (Mar. 2022); <http://dx.doi.org/10.1145/3516423>

Micah D. Beck (mbeck@utk.edu) is an associate professor at the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN, USA.

Terry R. Moore (tmoore@icl.utk.edu) is an associate director (retired) at Innovative Computing Laboratory, University of Tennessee, Knoxville, TN, USA.

The authors gratefully acknowledge the contribution of Elaine Wenderholm.

Copyright held by authors.

Viewpoint

Trust Is Not Enough: Accuracy, Error, Randomness, and Accountability in an Algorithmic Society

Searching for improved algorithmic accountability.

WE NOW TRUST many computer systems—hardware running deterministic, hard-coded software programs—to be much more reliable than they once were. And societally, we now have more of a sense of how to hold those who develop and operate software accountable when their products fail. Nevertheless, we are currently amidst a new set of concerns about the trust we grant to algorithmic systems—computational routines that “learn” from real-world data—to accomplish important tasks in our daily lives. While developers of algorithmic systems rightfully strive to make their products more trustworthy, and face room for improvement^{1,12} “trust” is an insufficient frame for the relationship between these technologies and the impacts they produce. Given the properties of machine learning and the social, political, and legal structures of accountability in which they are enmeshed, there currently unresolvable uncertainties about how such systems produce their results, and who should be held accountable when those results produce harm to individuals, communities, or all of society.

This uncertainty about how, whether, and when algorithmic harms come to pass is not ever going away, at least not completely. So, we remain in need of mechanisms to address

both *whom* to hold accountable and *how* to hold them accountable, rather than to rely on them to make themselves more trustworthy.

The harms for which accountability mechanisms are needed, and for which no responsible actor should be entrusted to minimize on their own, include harms that have been clearly defined by courts—injury, property loss, workplace hazards—and harms that have been difficult to pursue redress for in court—privacy violations,¹³ manipulative practices, and many forms of discrimination.

One reason it is so difficult to develop mechanisms of accountability is that such mechanisms speak to one of two complementary aspects of accountability without considering the other. One of these aspects of accountability is the role inhabited by those who *ought* to be accountable. In this role, accountable actors—engineers, corporate officers, or licensing bodies—must conduct themselves with a willingness to accept blame and with the capacity to see themselves as responsible to others for doing so. They must make themselves trustworthy



and accountable. The other aspect of accountability is the ability of others—harmed parties or those acting on their behalf—to hold responsible parties accountable, regardless of their willingness to accept blame. This second aspect of accountability requires social, institutional relations—between individuals, the organizations they might work for or in, and institutions that are themselves held accountable to the public—for accountability to exist particularly when those who ought to be trustworthy fall short.

These two aspects work together, even if accountability frameworks all too often do not. Personal and institutional forms of accountability reinforce each other. But because of the properties of algorithmic systems, they present significant barriers to accountability that go beyond those associated with accountability for traditional computational software systems. In our recent work,⁴ A.F. Cooper, Benjamin Laufer, Helen Nissenbaum, and I identify several of these barriers, and what follows is a discussion of how they work together to prevent individual actors from seeing themselves as accountable, and to prevent our social institutions—regulators, courts, companies, and the public—from holding anyone accountable when algorithmic systems produce harm. This Viewpoint extends this work to suggest immediate steps technical researchers, social scientists, and policymakers can take to make the world safer when the trust we place in developers falls short.

These barriers to accountability are characterized by the tendency of those who build and use algorithmic systems to shift the blame for algorithmic harms onto external, uncontrollable factors in order to recast harms as something for which no one could be seen as blameworthy. The means by which this barrier is erected is through scapegoating: attributing errors to fundamentally intractable statistical properties of the algorithmic system or to the fundamentally stochastic properties of the real world. In traditional software engineering, such errors are bugs, said to be “inevitable” and “endemic to programming.” Predictable in their unpredictability, bugs arise from errors in how

Uncertainty about how, whether, and when algorithmic harms come to pass is not ever going away.

engineers design, model, and code their software programs. But algorithmic systems present additional types of bug-like behavior to computational systems: misclassifications, statistical error, and nondeterministic outputs that may cause harm. Examples of these “bugs” abound: Google Photos labeling a photo of a Black couple as containing “gorillas,”⁵ translation and text-generation tools associating stereotypically gendered pronouns with various professions (for example, “he is a doctor,” “she is a nurse”), or grocery stores serving Muslim East African immigrant communities being flagged for fraud because their customers pooled their money to make large communal purchases.¹⁴ These errors are easily depicted in ways that deflect responsibility away from developers and onto the systems themselves, as the unavoidable and inevitable cost of rapid, but incremental progress.

Several factors contribute to these errors. One is the vested interests powerful institutions hold in allowing these mistakes, which uphold patterns of white supremacy, to go uncorrected. Beyond this is the practical challenge of collecting enough data representative of the phenomena developers are interested in. Many computer vision systems have been trained on datasets that underrepresent minoritized community members, which results in higher error rates for individuals from those communities.² Another challenge is mapping the available data appropriately onto the phenomena developers are interested in. Developers often must use proxies for the phenomena they are interested in, and so might use the cost of medi-

cal care as an approximation for the health needs of various communities without realizing the errors incurred by historical underinvestment in African American communities’ health-care.¹¹ These challenges are often seen as ultimately resolvable with better data or better problem formulations. But a third challenge is fundamentally unresolvable: many phenomena are stochastic. Many processes are too random for individual instances to be predicted. This is certainly true of physical processes such as alpha particle decay,⁷ but social processes, too, exhibit stochastic properties. When algorithmic systems are built to predict unpredictable phenomena, the error rate cannot be reduced beyond certain necessary thresholds.

To be sure, progress is being made—both rapidly and incrementally—to reduce the prevalence of such errors. But the impacts of imperfect systems already in use suggest parallel efforts are needed to address the inevitability of imperfection inherent in algorithmic systems. One of these efforts should strive to better evaluate whether a system is reliable enough to be used in the domain it is designed for. It stands to reason that in high-stakes domains where a great many people are expected to be affected, or a few people are expected to be affected in grave ways, the tolerance for error should be much lower than in trivial applications, and trusting the developer to draw the line is insufficient. But understanding what the stakes are in any given domain, and what the consequences of error are given those stakes, is no simple task.⁹ Nor is the prospect of providing reliable guarantees of robustness for systems’ behavior in-context.⁶ Qualitative methodologies developed in the social sciences can be adapted to understand the risks and equities for using algorithmic systems for specific purposes in specific places. These methods can support better decision making for those who must choose whether or not an application is “ready for prime time” before integrating it into people’s lives and communities.¹⁵ Proposed regulation in the E.U. and U.S. acknowledges the importance of this effort, stratifying compliance requirements by the risk profile and domain, and calling for

the prior study of algorithmic systems before they are deployed, in ways that incorporate community input and multidisciplinary expertise.

A second effort is required to make whole again those who have been harmed when a system errs. In this effort scapegoating constitutes the most serious barrier to accountability, as shifting blame for error toward an automated system shifts the responsibility to do something about that error away from those who are best positioned to offer a remedy. While requiring those who develop and operate algorithmic systems to document the expected impacts of their products go some distance toward establishing who the responsible parties are for any harmful errors,¹⁰ these requirements do not resolve the problem on their own. Those who are harmed when systems err also require meaningful recourse, and we can trust that companies that develop algorithmic systems are investing heavily in ways of limiting their own liability.³ Harmed parties must know who to contact, how to contact them, and what the circumstances of an adverse effect were in order to ask for a reconsideration or reversal of an algorithmic outcome. They also need to know a system was in use at all, which is not always readily apparent to users. Additionally, a private right of action⁸ would allow a means for redress when avenues of access to system operators is forestalled. A private right of action gives those who have been harmed by algorithmic systems standing in courts, to seek remedy for algorithmic injury, and to enter into discovery about the properties of systems that lead to their injury. The laudatory effects of such a right are two-fold—liability for harms is a powerful motivation for developers to resist deployed systems with unreasonably high likelihoods of erring in harmful ways, and it provides a compensatory avenue for those who have been hurt by harmful errors.

What the preceding suggests, most strongly, is that a new program of research is sorely needed. Developers are indeed making themselves more trustworthy.¹ But while significant progress is being made on developing algorithmic systems that are more accurate, and that sacrifice less accuracy when they pursue social and legal goals like fair-

The impacts of imperfect systems already in use suggest parallel efforts are needed to address the inevitability of imperfection inherent in algorithmic systems.

ness, the inherent stochasticity of such systems call for a better understanding of how irreducible errors are distributed across society. This means incorporating the above concerns—knowing when error rates are acceptable for a context of use and providing avenues to remedy for those who are harmed—into existing fields of computational research.

Doing so can be accomplished in three ways, each of which fills gaps created when trust fails.

► Increased audit and assessment throughout the machine learning pipeline can help to both clarify the relationship between design choices and resultant harms, as well as to identify approaches where error is irreducible and unsafe for use within specific contexts. Computational work on algorithmic robustness, in partnership with social scientists who understand the contexts of deployment is crucial for this.

► Rigorous standards of care can both motivate audit and assessment, but also can provide a rationale through which individual practitioners can see themselves as responsabilized to intervene when designs or deployments go awry.

► Lawmakers can enact legislation that enshrines a private right of action for those harmed by algorithmic systems. The technical expertise needed to demonstrate this harm can serve a dual purpose when it is applied prior to harm, making systems safer and limiting their use when not appropriate.

Current work on human-computer

interaction can also be expanded to include an emphasis on how to provide recourse to those who interact with algorithmic systems, to know that such systems are in use, and to contest adverse decisions. Both efforts require better and more practical documentation of algorithmic systems, which would lay out for all to see what the expected behaviors of algorithmic systems are, what contexts they are intended to be used in, what consequences might occur if used beyond their specifications, and who the relevant actors are who can review decisions, grant recourse, and be held accountable by others. Finally, the rise of computational superbugs requires a measure of modesty on the part of developers, to avoid rushing systems to market or making grandiose claims about their systems' capabilities, in favor of building systems that—when they err—err on the side of caution. 

References

1. Avin, S. et al. Filling gaps in trustworthy development of AI. *Science*. 374, (2021), 1327–1329.
2. Buolamwini, J. and Gebru, T. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Proceedings of Machine Learning Research* 18 (2018).
3. Burton, S. et al. Mind the gaps: Assuring the safety of autonomous systems from an engineering, ethical, and legal perspective. *Artificial Intelligence* 279, 103201 (2020).
4. Cooper, A.F. et al. *Accountability in an Algorithmic Society: Relationality, Responsibility, and Robustness in Machine Learning*. (2022); <https://bit.ly/3KZhdWY>
5. Dougherty, C. Google photos mistakenly labels black people "gorillas." Bits Blog (2015); <https://nyti.ms/2HuInr2>
6. Hancock-Li, L. Robustness in machine learning explanations: Does it matter? In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (ACM, Barcelona Spain, 2020); <https://bit.ly/4OptWYx>
7. Loveland, W.D. et al. *Modern Nuclear Chemistry*. John Wiley & Sons, Hoboken, NJ, 2006.
8. Mendelson, Stop Discrimination by Algorithms Act of 2021 (2021); <https://bit.ly/3LlsEta>
9. Metcalfe, J. et al. Algorithmic impact assessments and accountability: The co-construction of impacts. In *Proceedings of the ACM Conference on Fairness, Accountability and Transparency* (ACM, Toronto, ON, 2021); <https://bit.ly/4OocHqt>
10. Metcalfe, J. et al. A relationship and not a thing: A relational approach to algorithmic accountability and assessment documentation. *arXiv preprint*, 19 (2020).
11. Obermeyer, Z. et al. Dissecting racial bias in an algorithm used to manage the health of populations. *Science* 366, (2019), 447–453.
12. Shilton, K. et al. Excavating awareness and power in data science: A manifesto for trustworthy pervasive data research. *Big Data & Society* 8 (2021), 1–12.
13. Solove, D.J. A taxonomy of privacy. *University of Pennsylvania Law Review*. 154 (2006), 477.
14. Sloane, M. et al. *AI and Procurement: A Primer*. New York University, New York, (2021); doi:10.17609/BXZF-DF18.
15. Sloane, M. and Moss, E. AI's social sciences deficit. *Nat. Mach. Intell.* 1, 330–331 (2019).

Emanuel Moss (emanuel.moss@intel.com) is a Sociotechnical Systems Research Scientist at Intel Labs and former Postdoctoral Fellow at Cornell Tech and Data and Society Research Institute, New York, NY, USA.

Copyright held by author.

Viewpoint

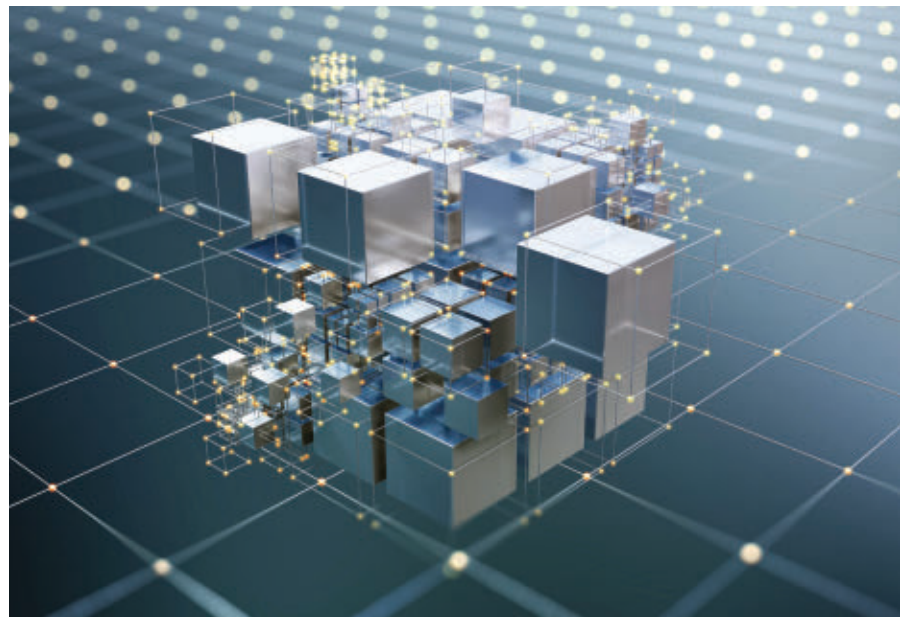
Decentraland: The Alleged Decentralization of Blockchain Applications

Breaking down the blockchain.

INSPIRED BY A post by Moxie Marlinspike on the subject of Web3,⁵ I wish to elaborate on the promise of decentralization of blockchain applications. I will focus on two major use cases of blockchain technology, namely finance and art, which beg the questions: What does decentralization mean? Are blockchain finance and art truly decentralized? And if not, what are the forces that drive us toward centralization?

I am aware of the issue of centralization of existing blockchains. However, this topic is abundantly covered in the previous literature (a comprehensive entry point is Sai⁶). What is missing, or at least less approached, is a discussion on the level of effective decentralization of blockchain applications, in particular of finance and art. Therefore, I focus this Viewpoint on these applications.

A Web3 marketplace is fundamentally different from a centralized Web2 application.



What Is Decentralization?

In information science, a centralized organization is typically represented by the tree data structure and commonly associated with the following principles:⁴

- *Centralism.* The tree structure has a unique root node from which all other nodes descend. This expresses a concentration of power and authority in a central actor of the system.

- *Finalism.* It describes the unidirectional, linear courses of trees: any point in a tree is reachable from the root by a unique, linear, top-down path of intermediate nodes. Trees therefore

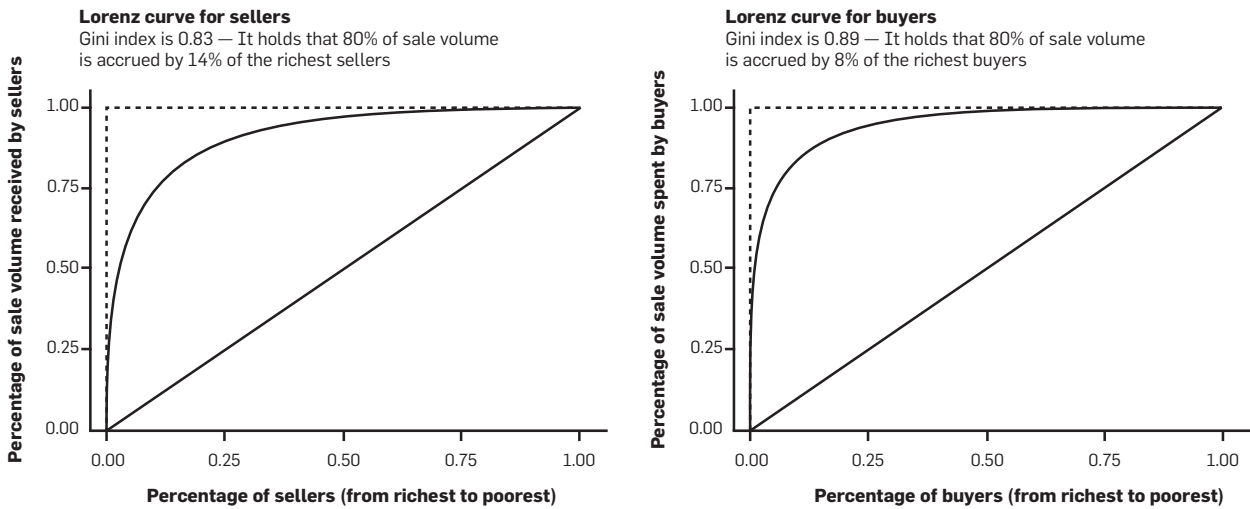
embody an organization devoid of multilinearity or feedback loops.

On the other hand, a decentralized organization is generally associated with the *network* data structure. Networks are rhizomatic structures defined by the following principles:⁴

- *Decentralization.* Networks are webs without a spider: there exists no central authority or root that regulates their growth, but they evolve in a self-organized way.

- *Multilinearity.* Networks allow feedback loops. This means that actors of the system can be reached through multiple paths. Rhizomatic (network)

Comparison of buyers and sellers.



structures offer a new model for knowledge and society that acknowledge decentralization, autonomy, flexibility, creativity, diversity, collaboration, altruism and, ultimately, democracy.

Networks match and sustain the proliferation of information typical of the postmodern condition—the co-existence of a multiplicity of heterogeneous discourses—instead of a simple, central discourse that unifies all forms of knowledge.¹

An important example of the modern perspectival shift from tree (centralization) to network (decentralization) is knowledge. Denis Diderot and Jean le Rond d'Alembert, in their *Encyclopédie* published between 1751 and 1772, depicted the genealogical structure of knowledge as a tree with three prominent branches—memory and history; reason and philosophy; and imagination and poetry—whereby only a small circle of scholars could participate as contributors. By contrast, the contemporary encyclopedia is incarnated by the knitted network of Wikipedia in which everyone can potentially contribute an article. As another example of this shift, corporations are evolving more fluid and dynamic models of decentralized autonomous organization (DAO) out of the typical monolithic hierarchies that have characterized business in the past.

My intention here is to investigate if and how the decentralization

promised by the cypherpunk origins of blockchain is still present in blockchain applications of finance and art—arguably the two major applications of digital ledger technology today.

Decentralization in Blockchain Finance

The financial crisis of 2008 led to the collapse of a number of major banks, such as Lehman Brothers, and forced many states to bail out others with public money. This crisis triggered an economic recession that has not yet been fully absorbed. On a psychological level, 2008 undermined the trust on which the relationship between citizens and their banks, and citizens and their states were based.

On Halloween of that fateful year, something else happened. A mysterious figure hiding behind the pseudonym Satoshi Nakamoto circulated an article, *Bitcoin: A Peer-to-Peer Electronic Cash System*, preceded by a statement

that left no doubt as to their motives: “I’ve been working on a new electronic cash system that’s fully peer-to-peer, with no trusted third party.”

However, despite the original motivations of fully peer-to-peer operations with no trusted third party, a large share of blockchain finance today is in fact *Centralized Finance* (CeFi), and not *Decentralized Finance* (DeFi). A CeFi platform uses blockchain technology but it is owned by a private company. Users must trust a third party to monitor transactions and ensure the security of assets. Such services also require one to submit one’s personal information for verification in order to operate (this is known as the Know Your Customer or KYC process).

The two pillars of blockchain finance are *stablecoins* and *exchanges*. The most popular stablecoins, like USDT and USDC, are centralized and collateralized by the U.S. dollar, which, like all fiat currencies, is centralized. USDT is owned by Tether, while USDC is owned by Centre, which was in turn founded by Circle and Coinbase. In contrast, DAI is the first decentralized stablecoin, maintained and regulated by MakerDAO, a decentralized autonomous organization that governs its parameters in order to ensure the stability of the stablecoin (which is pegged to the dollar). DAI is collateralized by Ether and other cryptocurrencies (including some centralized stablecoins), and not by fiat money.

**The two pillars
of blockchain finance
are stablecoins
and exchanges.**

Moreover, the most important cryptocurrency exchanges by trade volume, such as Binance, Coinbase, and Kraken, are centralized under private business identities. This effectively means that, when a customer moves funds on these exchanges, the funds are deposited in the exchange's wallet, and the customer does not own them anymore in their wallet. The customer is forced to trust the third party, as one would when depositing money with a traditional bank. All transactions cost a fee that goes directly to the exchange.

On the other hand, on a decentralized exchange such as Uniswap, customers rely on smart contracts in order to operate (to swap coins or deposit liquidity). Smart contracts are public, audited by third parties, and deposited on the blockchain. Hence, they always do what they are supposed to do (including executing bugs), and cannot be altered unless a new contract is deployed according to the will of a community of token holders. Market takers (those who swap coins) pay a fee that remunerates the market makers (the providers of liquidity). Notably, when a user swaps coins, the exchanged funds are deposited directly into their wallet. In this case, users are forced to trust not a third party but the exchange developers and the smart contracts they wrote, although the code is open source and hence can be audited by anyone.

Decentralization in Blockchain Art

Crypto art is a recent artistic trend leveraging on the novel idea to make a digital file scarce by associating it with a non-fungible token or NFT on the blockchain.³

Despite crypto art's original commitment to decentralization and equal distribution of resources, the crypto art market is now highly concentrated around a small number of artists and collectors. On SuperRare (one of the major crypto art marketplaces), I computed that 80% of the sale volume (between April 2018 and January 2022) was made by 14% of the top artists (Gini index equal to 0.83) and collected by 8% of the top collectors (Gini index of 0.89). By comparison, in the U.S., the Gini index of household incomes was 0.48 from 2014 to 2018. On other

Not all individuals have the will, skill, and time to adapt to a truly interdisciplinary field such as blockchain.

marketplaces, such as Foundation, the situation is similar or even worse.⁷ In other words, if crypto art were a country and sales reflected a population's income, then wealth would be as concentrated as it is in the wider economy.

Another point of concentration concerns marketplaces. In principle, an artist can write and deploy on the blockchain their own smart contract and use it in a pure peer-to-peer fashion to interact with collectors and sell their art. In practice, few artists have these technical skills and prefer to use smart contracts developed by marketplaces. In all cases, only the NFT is deposited on the blockchain (hence fully decentralized), while the artwork's metadata and media containing the actual digital art are typically stored on either a centralized server such as Amazon Web Services (AWS), or on some nodes of a peer-to-peer network such as IPFS and Arweave. The transaction minting the NFT on the blockchain records (but does not verify) the link between the token and the metadata and artwork files.

That said, a Web3 marketplace is fundamentally different from a centralized Web2 application. The recent case of Hic et Nunc (a popular crypto art gallery on the Tezos blockchain) exemplifies this. When its creator, Rafael Lima, decided to shut down what was a highly successful NFT marketplace—not to mention a progressive ecosystem based on the Tezos blockchain—many users felt scammed and some even claimed it was a rug pull. Yet only the front end of the website disappeared, while all the smart contracts and NFTs were still immortalized on the blockchain and visible via other NFT aggregators, like Objkt.com. Now

Hic et Nunc is fully functional under the new name Teia.

Centralization Is Delegation, Decentralization Is Responsibility

Moxie Marlinspike, the founder and CEO of messaging app Signal, recently addressed the effective decentralization of Web3.⁵ In his view, the Web3 architecture follows the client-server model but virtually all clients who wish to access it do so by trusting the outputs of servers run by a small number of companies including Etherscan, Infura, and OpenSea. However, this inevitably leads to some form of centralization because:

- ▶ people, even “nerds,” do not want to run their own servers and never will;
- ▶ the burden of building software is high. At this point, software projects require an enormous amount of human effort;
- ▶ a protocol such as the blockchain moves much more slowly than a platform such as OpenSea. If something is truly decentralized, it becomes very difficult to alter, and often remains stalled in time.

As Marlinspike hinted in his post, not all individuals have the will, skill, and time to adapt to a truly interdisciplinary field such as blockchain. A dose of delegation and centralization is therefore inevitable. This is not a bad thing, so long as we deeply understand that centralization entails delegation, while decentralization demands responsibility. 

References

1. Cilliers P. *Complexity and Postmodernism: Understanding Complex Systems*. Routledge, London and New York, 1998.
2. Franceschet, M. Are blockchain art and finance really decentralized?; <https://bit.ly/41K7yu2>
3. Franceschet, M. Crypto art: A decentralized view. *Leonardo* 54, 4 (2021).
4. Lima, M. *Visual Complexity: Mapping Patterns of Information*. Princeton Architectural Press, New York, 2011.
5. Marlinspike, M. My first impressions of Web3; <https://bit.ly/445hJfI>
6. Sai, A.R. Taxonomy of centralization in public blockchain systems: A systematic literature review. *Information Processing & Management* 58, 4 (2021).
7. Vasan, K. et al. Quantifying NFT-driven networks in crypto art. *Scientific Reports* 12 (2022).

Massimo Franceschet (massimo.franceschet@uniud.it) is an associate professor in the Department of Mathematics, Computer Science and Physics at the University of Udine, Italy.

This Viewpoint is an excerpt of a longer post by the author.² The author thanks Alex Estorick for editing the original version of this Viewpoint.

Copyright held by author.

The case for in-memory inferencing of quantized CNNs at the edge.

BY GABRIEL FALCAO AND JOÃO DINIS FERREIRA

To PiM or Not to PiM

AS ARTIFICIAL INTELLIGENCE becomes a pervasive tool for the billions of Internet of Things (IoT) devices at the edge, the data movement bottleneck imposes severe limitations on the performance and autonomy of these systems. Processing-in-memory (PiM) is emerging as a way of mitigating the data movement bottleneck while satisfying the stringent performance, energy efficiency, and accuracy requirements of edge imaging applications that rely on convolutional neural networks (CNNs).

The globalization of affordable Internet access has spurred a revolution in computer architectures, characterized by the accelerate widespread adoption of smartphones, tablets, and other smart devices, which are now commonplace.¹⁵ The rise of IoT applications in a wide range of domains (for example, personal computing, education, industry, military, healthcare, digital agriculture) brings with it the ability to integrate billions of devices on the Internet, as depicted in Figure 1.

This integration presents unprecedented challenges, such as the need for inexpensive computation and communication, capable of crunching the increasing volumes of data generated every day.

The IoT paradigm also promises a more intimate connection between the cyber and physical worlds, as data becomes a ubiquitous asset exchanged among all manner of connected (smart) devices. Moreover, the data flow is often bidirectional, taking place not just from the physical to the cyber world, but also from the cyber to (what is possible in) the physical world.

Bringing this vision to fruition will require that IoT devices exhibit AI. The most promising approaches today are based on empowering systems with the ability to learn autonomously from experience by assimilating large amounts of data—using machine learning (ML) algorithms—with a particular focus on deep learning and image inference.

AI Demands at the Edge Will Grow

The total volume of digital data created, replicated, and consumed within a year has surpassed dozens of zetabytes (ZBs or 10^{21} bytes) in 2020, and the International Data Corporation estimates that this number will grow to hundreds of ZBs in coming years.¹⁸ The COVID-19 pandemic contributed to this figure because of widespread work-from-home mandates and a sudden increase in videoconferencing and streaming data. A significant portion of this data is consumed at the edge, often with processing performed entirely



on smartphones and embedded systems. The rise of many other AI-based applications applied to the big data revolution exacerbates this problem by placing increasing stress on computing and memory systems, particularly those operating at the edge.

The case for edge computing as the enabler of sustainable AI scaling is strengthened because a sizable portion of the data generated by modern digital systems originates from sensors located at the edge—under this paradigm, data is processed where it is generated, as Figure 2 illustrates. In contrast with the conventional approach shown on the left, with the emergence of cloud-edge hierarchies, AI moves to the edge layer, shown on the right in the figure. This new paradigm creates pressure for more inten-

sive computation on the edge-processing nodes, but it also decreases the time and energy spent communicating with the cloud and introduces data privacy and latency benefits.

With the growth in data generation and the expansion of demand for AI applications at increasing rates, five goals justify moving the processing of AI to the edge.^{1,6,20,24}

- **Latency.** Many applications have an interactive nature and thus cannot endure long latencies, especially for memory, storage, and network requests.

- **Reliability.** Communication networks are not reliable at all places, at all times. To ensure maximum uptime, AI-based decision making must not rely on always-available communication networks.

- **Privacy/security.** Some applications

require sensitive data to be kept in a controlled local environment, avoiding its circulation to/from the cloud. Examples are medical, finance, or autonomous driving, among many others.

- **Bandwidth.** Data that is processed near where it is collected does not need to be sent to the cloud, which reduces the overall bandwidth demand on the network and the edge-computing systems.

- **Data provenance.** Provenance issues may prevent data from being processed far from where it is generated. Datacenter storage may need to comply with regional data protection legislation such as General Data Protection Regulation (GDPR) in Europe and Personal Information Protection Act (PIPA) in Canada.

To achieve these goals, computer ar-

Figure 1. IoT vs. non-IoT active device connections.

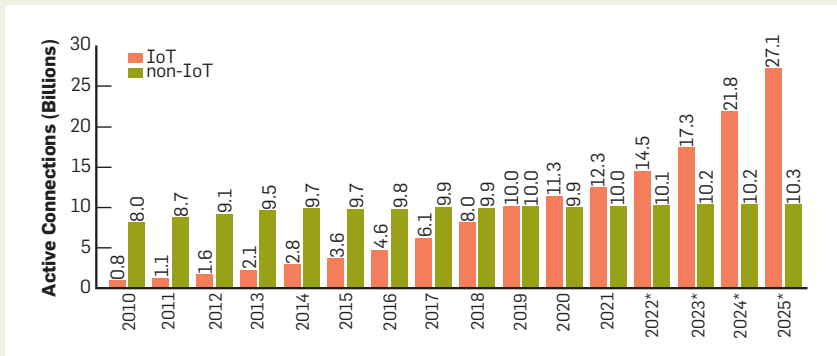


Figure 2. Migration of AI to the edge.

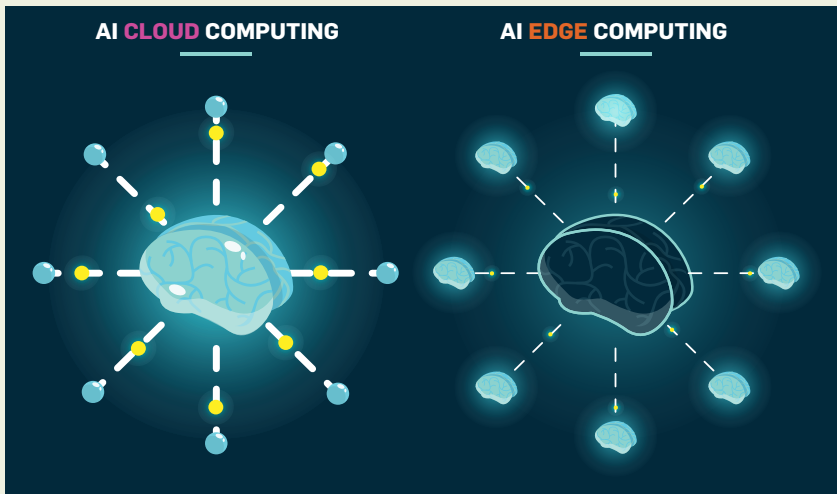
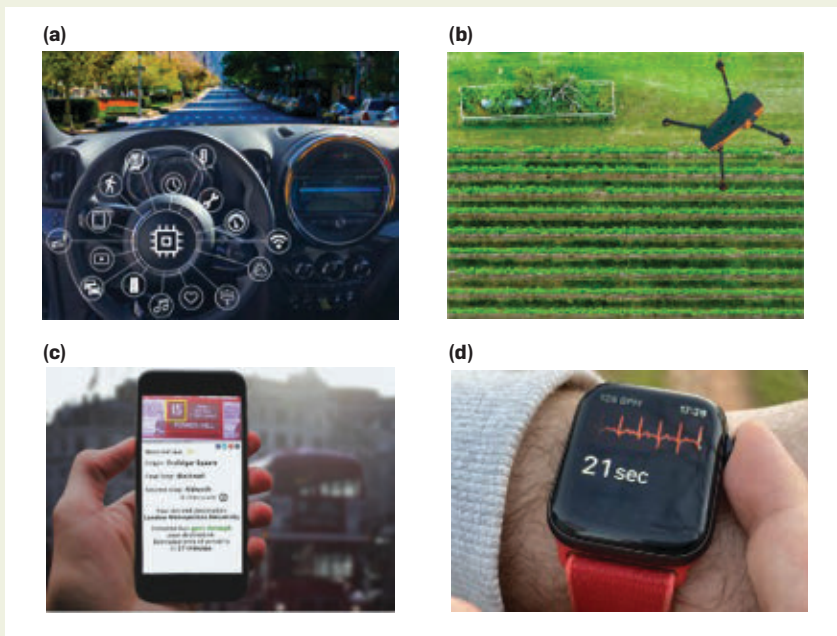


Figure 3. Examples of data processing with AI kernels at the edge.



chitects and software developers must adopt a holistic vision of the combined cloud+edge system to keep the unnecessary movement of data between components to a minimum by processing data where it is generated and stored, as this is the dominant performance and energy bottleneck.⁹

AI at the Edge: ML Solutions for Data Challenges

At present, neural networks are widely used in many domains and are becoming integral components of other emerging applications, such as self-driving cars, always-on biosignal monitoring, augmented and virtual reality, critical IoT, and voice communication (which represents up to 25% of the use cases of 5G at the edge), all of which require AI algorithms to operate on high volumes of data at the edge (see Figure 3). Examples include autonomous vehicles, digital agriculture, smartphones, and smart IoT devices that process substantial volumes of data while running AI kernels at the edge. The systems that support these emerging applications will be expected to make decisions faster—and, often, better—than their human counterparts, with support for continuous fine-tuning of their decision-making by factoring in ever-increasing volumes of data for training and inference.

CNNs are the de facto standard for image-based decision-making tasks. These models make heavy use of the convolution and multiply-and-accumulate (MAC) operations, which represent more than 90% of the total cost of computation.¹¹ For this reason, state-of-the-art neural network accelerators (for example, Google's Tensor Processing Unit) have focused on optimizing the performance and energy efficiency of MAC operations.

Characterizing and optimizing CNN architectures. Typical CNNs consist of hundreds of millions of parameters and require the execution of billions of operations per second to achieve real-time performance. As their accuracy improves, CNNs include more parameters and layers, becoming wider and deeper. The use of compact data representations provided by quantization mitigates some of the overhead of these more complex network architectures and allows for high degrees of

parallelism and data reuse, which are especially useful in constrained processing environments.

Data quantization significantly reduces the computation and storage requirements of neural networks by decreasing the bit width of the model's weights. Quantizing these values to less than 8 bits while retaining accuracy, however, requires manual effort, hyperparameter tuning, and intensive retraining.

While training requires a high dynamic range, inference does not: In most cases, 2-bit to 4-bit precision achieves the desired levels of accuracy.¹⁰ Going further, it is even possible to approximate CNNs by binarizing (that is, quantizing to one bit) their input, weights, and/or activations.^{10,17}

In many practical settings, particularly edge computing, the performance and energy-efficiency benefits of binary neural networks outweigh the accuracy loss. A further benefit of binary neural networks is the ability to approximate the convolution operation required by CNNs, by combining the much more efficient bitwise XNOR and bit-counting operations.

Table 1 illustrates the memory and processing requirements of five widely used CNN models across a range of devices, from datacenter servers to IoT nodes. The table indicates the number of parameters and MAC operations for all the networks. W/A refers to the bit widths of weights and activations for each level of quantization. All networks have 224-by-224 input resolution. These devices were selected from a large set of neural network algorithms to fit the edge nodes, namely in terms of memory required to run the models. Their size is shown as a function of weight/activation bit widths (32-bit, 2-bit, 1-bit), as well as the number of MAC operations used by each network.

Specialization as the enabler of high-performance AI. Modern deep-learning algorithms have substantial computational, memory, and energy requirements, which makes their implementation on edge devices challenging. This challenge can be addressed by exploiting two unique characteristics of ML algorithms: First, each class of deep-learning algorithm relies on a limited set of *specialized operations*; second, in many cases these algorithms provide

good accuracy even when they use low-bit-width operations.¹³

In recent years, several frameworks (for example, TensorFlow, PyTorch, TensorRT) helped bridge the semantic gap between the high-level description of a neural network model and its underlying hardware mapping by using specialized instructions. This is achieved by performing operations in a bulk parallel manner, minimizing memory accesses and maximizing compute resource utilization.

Efficient Edge AI: Architecting Data-Centric Systems

These observations and constraints lead to the formulation of a set of well-defined target metrics for AI at the edge:

- **Accuracy.** The success rate of the AI task¹⁰ (for example, image classification, object detection, sentence generation, translation).

- **Throughput.** The rate of processing of input data. Many real-time AI applications that support video must sustain processing rates on the order of thousands of FPS (frames per second)—for example, 2,300FPS in self-driving cars²³ or hundreds to thousands of FPS in ultrasound medical devices.

- **Latency.** The critical path delay associated with the processing of a single input element. 5G standards define a maximum latency of one millisecond for positioning and tracking systems; self-driving cars must provide latencies within the same order of magnitude.²³

- **Power and energy.** Most edge devices are battery-powered and maximizing battery life is a key design target. For reference, the computational system of self-driving cars requires a power supply of up to 2.5kW.²³

- **Data precision.** AI data need not always be represented in 64- or 32-bit floating-point precision. For many in-

ference applications, integer precision of eight bits or less suffices.^{4,13}

The good old processor-centric computing model. The late 20th and early 21st centuries saw the widespread use of the processor-centric computational model. In this model, programs and data are stored in memory, and processing takes place in specialized ALUs (arithmetic logic units). Together with Moore's Law, the introduction of caches, branch predictors, out-of-order execution, multithreading, and several other hardware and software optimizations enabled a steady and mostly unfazed series of performance improvements over the past decades.

In contrast, memory systems have improved at a much slower pace. This performance gap between the processor and the main memory—supported by the fact that these technologies are still several process-node generations apart—has given rise to a critical data movement bottleneck, dubbed the *memory wall*.¹⁴ Memory is the dominant performance and energy bottleneck in modern computing systems; data movement is much more expensive than computation, both in latency and energy. The data movement bottleneck will remain relevant as the number of smart devices connected to the Internet—as of this writing, already in the billions, as depicted in Figure 1—continues to grow.

A compelling possibility: processing data where it resides. As the demand for inferencing at the edge grows, accessing data more efficiently becomes increasingly relevant. The proposed improvements span data reuse by exploiting temporal and physical locality; algorithm design, with the introduction of optimized neural network topologies and the use of quantization; specialized hardware, introducing dedicated vectorized instructions

Table 1. Convolutional neural networks using the Imagenet dataset.

Neural Network	Size 32/32 (W/A) (MB)	Size 2/2 (W/A) (MB)	Size 1/1 (W/A) (MB)	#Params. (M)	#MAC operations (G)
AlexNet (2012)	237	14.8	7.4	62	0.7
MobileNet-v2 (2018)	13	0.8	0.4	3.4	0.3
ResNet-18 (2015)	45	2.8	1.4	12	1.8
VGG16 (2014)	515	32.2	16.1	135	15.5
Inception-v3 (2015)	91	5.6	2.8	24	2.8

designed to address the demands of these workloads (for example, SIMD MAC operations); and PiM architectures. When performed at the edge, PiM enables higher throughput for AI applications, without compromising device autonomy.¹⁷

PiM solutions differ primarily in their proximity to the data:¹⁴ In the processing-near-memory (PnM) paradigm, computation takes place close to where data resides, but in a different medium—for example, in the logic layer of a 3D-stacked memory; in contrast, the processing-using-memory (PuM) paradigm takes advantage of the storage medium's physical properties to perform computation—for example, ReRAM (resistive random access memory) or DRAM (dynamic random access memory) cells.

Processing-near-memory. 3D-stacked memories are an emerging type of memory architecture that enables the vertical stacking of memory layers on top of a logic layer. This logic layer can be designed to feature hardware support for several operations, thus enabling computation inside the memory units.¹⁹

Processing-using-memory. DRAM technology is especially well suited for supporting bitwise operations since adjacent memory rows can communicate with one another through their bitlines.

Ambit²¹ supports bulk bitwise majority/AND/OR/NOT functions by exploiting the analog operation of DRAM. The combination of these operations allows the design of full applications. Recent studies show that Ambit's core operating principle can be performed in commodity off-the-shelf DRAM chips with no changes to DRAM.⁷ Ambit improves performance by 30 to 45 times and reduces energy consumption by 25 to 60 times for the execution of bulk bitwise operations, resulting in an overall speedup in database queries of four to 12 times.

SIMDRAM⁸ creates an optimized graph representation of a user-defined arbitrary operation using bitwise majority and *not* operations, which can be performed using the triple-row activation command defined in Ambit. The SIMDRAM control unit orchestrates the computation from start to finish by executing the previously defined DRAM commands. SIMDRAM improves per-



As AI becomes a pervasive tool for billions of IoT devices at the edge, the data movement bottleneck imposes limitations on these systems' performance and autonomy.



formance by 88/5.8 times and reduces energy consumption by 257/31 times, compared with CPU/GPU execution.

The DRAM-based PuM architecture pLUTo⁵ extends the flexibility and performance of PuM by introducing a mechanism for bulk in-DRAM value lookups. The lookups take place entirely within the DRAM subarray and therefore do not require that data be moved off-chip at any point. With pLUTo, it is possible to implement arbitrarily complex functions as table lookups (so long as the memory arrays are sufficiently large to accommodate them), while minimizing the overall movement of data. pLUTo improves performance by 33/8 times and reduces energy consumption by 110/80 times compared with CPU/GPU execution.

PRIME² and ISAAC²² are two promising neural network accelerators based on ReRAM. These proposals leverage the ReRAM crossbar array to perform matrix-vector multiplication efficiently in the analog domain. These solutions report performance and energy consumption improvements of up to 2,360 and 895 times, respectively, relative to state-of-the-art neural-processing unit designs.

Enabling the adoption of PiM. Because quantized neural networks rely on simple operations, they are a prime target for exploiting the use of PiM to process AI kernels at the edge. Performing more straightforward computation close to where data resides greatly reduces the overall movement of data, which improves latency, throughput, and energy efficiency. These can likely be applied even more efficiently to quantized and binary CNNs, which use XNOR and bit count operators.

To validate the performance and energy merits of PiM, we present a quantitative analysis of the improvements to CNN image inference at the edge, offered by Ambit, a DRAM-based PuM architecture that supports bitwise logic operations.

To PiM or Not to PiM: A Quantitative Analysis

This section evaluates the accuracy, performance, and energy costs of performing inference for binary and quantized versions of the most recent neural networks, introduced in Table 1. To this end, Table 2 shows the ener-

gy consumption of Ambit-based PiM designs to perform inference on each of these networks. These estimates account for the cost of performing the dominant MAC operations required by the convolutions. Energy is calculated for execution on both ARM CPUs at the edge (baseline), as well as using an analytical model for PiM technology. (For full access to the original references, see <https://github.com/joaddinissf/qcnn-accuracy/>.)

This shows that different accuracy-performance-energy trade-offs are possible. Table 3 is an analysis of the performance of the four most recent neural networks from Table 1, targeting 15FPS and 60FPS, for 3-, 2- and 1-bit precision, with supported infer-

ence throughput (in FPS) for Qualcomm Snapdragon 865, Intel Xeon Gold 6154, and Edge TPU baselines, compared with an Ambit-based PiM architecture. Cells with a ‘-’ indicate lack of real-time support; cells with a ‘+’ indicate real-time support. Empirical observations show that these quantized models ensure adequate accuracy for many applications. Ingenious quantization techniques enable further compression of even the smallest neural network models and the portability of very large networks for use in power-constrained devices, with small, tunable, accuracy losses.

Accuracy. The average accuracy loss is 6.8%, and in the best case the loss is as low as 2.2%. In cases where the ac-

curacy of the 1-bit network may result in too great of an accuracy loss, settling for 2- or 3-bit quantization often yields good accuracy values.

Performance. The described Ambit-based accelerator achieves 15FPS in eight cases out of 12, and 60FPS for the other four cases. In contrast, the Qualcomm Snapdragon, Intel Xeon, and Edge TPU baselines sustain 15FPS in, respectively, zero, three, and nine cases out of 12, and 60FPS in zero, zero, and nine cases out of 12. The average speedup of the Ambit-based accelerator over each of these baselines is 58.8, 21.3, and 4.6 times.

Two key conclusions can be drawn. First, the Edge TPU can sustain a processing rate of only 3.4FPS for VGG-16,

Table 2. Energy consumed per inference in Table 1.

Neural Network	Baseline Accuracy (%)			Energy of PiM Designs (mJ)			
	Top-1% 32/32 (W/A)	Top-1% 2 ≤ W/A ≤ 8	Top-1% 1/1 (W/A)	Baseline* 32/32	PiM 3/3	PiM 2/2	PiM** 1/1
AlexNet	60.0	58.2	57.0	344	173	91	10
MobileNet-v2	72.0	—	54.4	91	46	24	3
ResNet-18	69.1	67.0 [†]	64.8	1090	548	288	31
VGG16	71.1	69.1	68.9	9390	4716	2483	265
Inception-v3	78.0	76.4 ^{††}	—	3450	1734	913	97

* Qualcomm Snapdragon 865 with 4 ARM Cortex-A77 and 4 A55 CPUs using multilevel cache and DRAM-based main memory.

** Assumes 1-bit quantization of the neural network’s weights and activations and that all operations are performed in-memory on an LPDDR4 with clock frequency 2133MHz and tRCD = tCL = tRP = 15ns.

† Employs ternary data representation for both weight and activation values.

†† Employs 4-bit weights and 8-bit activations. The weights of the first and last layers are 8-bit values.

Table 3. Supported inference throughput for three baselines and PiM.

Neural Network	Bit Width	Able to achieve 60 FPS?				Able to achieve 15 FPS?			
		Snapdragon	Intel Xeon	Edge TPU	PiM	Snapdragon	Intel Xeon	Edge TPU	PiM
VGG-16	3	—	—	—	—	—	—	—	—
	2	—	—	—	—	—	—	—	+
	1	—	—	—	+	—	—	—	+
MobileNet-V2	3	—	—	+	—	—	+	+	—
	2	—	—	+	—	—	+	+	+
	1	—	—	+	+	—	+	+	+
ResNet-18	3	—	—	+	—	—	—	+	—
	2	—	—	+	—	—	—	+	+
	1	—	—	+	+	—	—	+	+
Inception v3	3	—	—	+	—	—	—	+	—
	2	—	—	+	—	—	—	+	+
	1	—	—	+	+	—	—	+	+

because of this network's high number of parameters. This illustrates the poor scalability of current specialized neural network accelerators to process very large networks. Second, the flexible degree of parallelism in the PiM implementation, attained through the operation of multiple subarrays in parallel, allows the inference time to scale quasi-linearly with the size of the network, which enables a near-constant inference time in the largest networks.

Energy. Average energy savings of 35.4 times were observed for 1-bit, relative to 32-bit precision. Energy gains are linearly proportional to the degree of quantization.

Open Challenges

As the volume of data to process approaches the installed processing capacity, the successful implementation of AI at the edge will depend on the development of optimized architectures that are able to perform AI tasks while meeting strict performance and energy-efficiency requirements. After the introduction of 1,000-core processors and the expansion of levels of parallelism in the compute units, computer architects and software developers must now turn to the memory to design the next generation of high-performance and highly efficient systems. While PiM architectures have shown promise,¹⁶ enabling their adoption at the edge layer will require answers to the following open questions for manufacturers designing PiM:

- *Software/hardware co-design.* The integration of PiM technology and its compilers with new firmware and the operating system should be facilitated to enable the design of edge systems with maximal performance and energy efficiency in order to meet the computational demands of AI applications and use cases. PiM can be specialized for the edge by supporting common image-processing operations, which are especially relevant for performance.

- *Low design complexity and cost.* PiM designs should be made sufficiently low-cost to entice hardware manufacturers to integrate them into their products. This will entail the development of early-access commercial prototypes and proof-of-concept products that encourage early adopters to

Smart devices may soon come to incorporate both conventional and PiM-enabled memories.

disseminate the technology and offset R&D costs. PiM technology is reaching a point of maturity, and it will soon make its way to self-driving cars, healthcare, digital agriculture, and other edge AI applications with massive total addressable markets.

- *Multitiered PiM architectures.* As PiM technology matures, it will populate multiple levels of the existing memory hierarchy with complementary processing components, each with its own benefits and drawbacks.^{3,12} For example, processing-in-cache architectures have been proposed, which trade off the DRAM capacity for greater speed and support for a wider range of operations. Analogously, processing-in-storage (typically oriented toward nonvolatile memories) is well suited for the execution of simpler operations on vast volumes of data, with very high throughput.

- *The compiler.* PiM substrates will require custom compilers; these compilers can be a tool to aid software developers in the identification of common memory access and computational patterns, in order to automate certain steps of the compilation process to yield maximal performance (for example, the adoption of ideal data mapping or the circumvention of bottlenecking memory accesses).

PiM software development for end users includes:

- *Algorithm design.* Mapping existing and emerging applications to PiM substrates requires the algorithm to exploit data quantization, exposed parallelism, and the coalescing of memory accesses to offset the high-cost operations in the system, which are especially costly at the edge. The programmer is responsible for achieving the desired trade-off between performance, energy efficiency, and accuracy to satisfy the requirements of the target AI application.

- *The development framework.* A low-cost approach to empower the compiler with knowledge about the application's data flow is to create an intuitive and expressive API that abstracts as many as possible of the high-efficiency operations supported by the PiM substrate in an easy-to-use way. It is tempting to draw an analogy between the current state of PiM and the early days of general-purpose computing on graphics processing units (GPGPU):

CUDA and OpenCL were instrumental in enabling the mass-scale adoption of GPUs, and a similar API must assume that role for PiM.

► **Benchmarking tools.** Standard benchmarking, profiling, simulation, and analysis tools enable the comparison of different architectures and are therefore essential for the R&D stage of PiM. This is especially important for emerging applications, as is the case for many edge AI workloads. Like the role that MLPerf (the result of a collaboration of a consortium of AI leaders from academia, research labs, and industry) plays in the development of machine-learning algorithms, a set of standardized tests would allow the development of PiM to remain steadfast and open.

What the Future Will Bring and the Role of PiM

Demand for AI at the edge will continue to grow. High-performance and energy-efficient PiM-based edge architectures for the processing of AI should have the following qualities:

► **Support quantized data structures.** As shown exhaustively in the literature, particularly for inference, the use of reduced-bit-width representations yields only small losses in accuracy. This is particularly useful for reducing the movement of data within/from/to the memory subsystem and for increasing the degree of vectorization and parallelism.

► **Support specialized instructions.** Bitwise and LUT (lookup table)-based operations can be implemented in the memory substrate as row-level operations or in the logic layer, with sizable benefits for running AI kernels with low hardware complexity, high bandwidth, and high parallelism.

► **Avoid the unnecessary movement of data.** The compiler should be able to detect what part of the AI kernel should run on the processor and what part should run in memory. Not only are different portions of the kernel more suitable for distinct subsystems, but it is also fundamental to balance the workload among them to maximize performance. The minimization of energy consumption is another target that the computer architecture community should aim for.

► **Foster a PiM ecosystem.** Achieving critical mass in the adoption of PiM is a crucial milestone for its success, and reaching it requires a programmer-

friendly interface, intuitive compilers, and comprehensive test suites, with a set of industry-standard benchmarks.

Where do we go from here? Several PiM architectures have demonstrated the ability to perform AI tasks with unprecedented levels of efficiency by meeting the take-aways presented here. Existing PiM designs support a limited range of operations, however, and further work is necessary to meet the requirements of AI tasks.

PiM is not a silver bullet, and it will not supersede conventional computing—although it may soon become evident that computers and other smart devices benefit from incorporating both conventional processing units and PiM-enabled memories. When designing new architectures, computer architects must remain mindful that data movement is expensive in both latency and energy. Emerging technologies and architectures enable the mitigation of data movement and, as such, pave a path for the design of more efficient computing devices. The current paradigm will take us only so far; PiM presents a compelling alternative.

Acknowledgments. This work was partially supported by Instituto de Telecomunicações and Fundação para a Ciência e a Tecnologia, Portugal, under grants EXPL/EEI-HAC/1511/2021, PTDC/EEI-HAC/30485/2017 and UIDB/EEA/50008/2020. C

References

- Bonawitz, K., Kairouz, P., McMahan, B., Ramage, D. Federated learning and privacy: building privacy-preserving systems for machine learning and data science on decentralized data. *acmqueue* 19, 5 (2021), 87–114; <https://dl.acm.org/doi/10.1145/3494834.3500240>.
- Chi, P., Li, S., Xu, C., Zhang, T., Zhao, J., Liu, Y., Wang, Y., Xie, Y. PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory. *ACM SIGARCH Computer Architecture News* 44, 3 (2016), 27–39; <https://dl.acm.org/doi/10.1145/3007787.3001140>.
- Devaux, F. The true processing in memory accelerator. In *Proceedings of the IEEE Hot Chips 31 Symp.* IEEE Computer Society, 2019, 1–24; <https://ieeexplore.ieee.org/document/8875680>.
- Duarte, P., Tomas, P., Falcao, G. SCRATCH: An end-to-end application-aware soft-GPGPU architecture and trimming tool. In *Proceedings of the 50th Annual IEEE/ACM Intern. Symp. Microarchitecture*, 2017, 165–177; <https://ieeexplore.ieee.org/document/8686531>.
- Ferreira, J.D. et al. pLUTo: Enabling massively parallel computation in DRAM via lookup tables. In *Proceedings of the 55th IEEE/ACM Intern. Symp. Microarchitecture*, 2022, 900–919; <http://bit.ly/318RCKZ>.
- Fuketa, H., Uchiyama, K. Edge artificial intelligence for the cyberphysical systems era. *Computer* 54, 1 (2021), 84–88; <https://ieeexplore.ieee.org/document/9321799>.
- Gao, F., Tziantzioulis, G., Wentzlaff, D. ComputeDRAM: In-memory compute using off-the-shelf DRAMs. In *Proceedings of the 52nd Annual IEEE/ACM Intern. Symp. Microarchitecture*, 2019, 100–113; <https://dl.acm.org/doi/10.1145/33552460.3358260>.
- Hajinazar, N. et al. SIMDram: A framework for bit-serial SIMD processing using DRAM. In *Proceedings of the 26th ACM Intern. Conf. Architectural Support for Programming Languages and Operating Systems*, 2021, 329–345; <https://dl.acm.org/doi/10.1145/3445814.3446749>.
- Hennessy, J.L., Patterson, D.A. A new golden age for computer architecture. *Commun. ACM* 62, 2 (Feb. 2019), 48–60; <https://dl.acm.org/doi/10.1145/3282307>.
- Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *J. Machine Learning Research* 18, (2017), 1–30; <https://www.jmlr.org/papers/volume18/16-456/16-456.pdf>.
- Jouppi, N. et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the ACM/IEEE 44th Annual Intern. Symp. Computer Architecture*, 2017, 1–12; <https://ieeexplore.ieee.org/abstract/document/8192463/similar>.
- Kwon, Y.-C. et al. A 20nm 6GB function-in-memory DRAM, based on HBM2 with a 1.2 TFLOPS programmable computing unit using bank-level parallelism, for machine learning applications. In *Proceedings of the 2021 IEEE Intern. Solid-state Circuits Conf.*, 350–352; <https://ieeexplore.ieee.org/document/9365862>.
- Marques, J., Andrade, J., Falcao, G. Unreliable memory operation on a convolutional neural network processor. In *Proceedings of the IEEE 2017 Intern. Workshop on Signal Processing Systems*, 1–6; <https://ieeexplore.ieee.org/document/8110024>.
- Mutlu, O., Ghose, S., Gómez-Luna, J., Ausavarungrun, R. A modern primer on processing in memory. 2020; <https://arxiv.org/abs/2012.03112>.
- Pandey, P., Pompili, D. Handling limited resources in mobile computing via closed-loop approximate computations. *IEEE Pervasive Computing* 18, 1 (2019), 39–48; <https://ieeexplore.ieee.org/document/8705029>.
- Radjokovic, P., Carpenter, P., Esmaili-Dokht, P., Cimadomo, R., Charles, H.-P., Sebastian, A., Amato, P. Processing in memory: The tipping point. White Paper from European Technology Platform for High-performance Computing, 2021; <https://bit.ly/3USrm8H>.
- Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A. Enabling AI at the edge with XNOR-networks. *Commun. ACM* 63, 12 (Dec. 2020), 83–90; <https://dl.acm.org/doi/10.1145/3429945>.
- Reinsel, D., Gantz, J., Rydning, J. The digitization of the world—from edge to core. I.D. Corp. White Paper, 2018; <https://bit.ly/40H3IM1>.
- Rosenfeld, P. Performance exploration of the hybrid memory cube. Ph.D. dissertation. University of Maryland, 2014; <https://user.eng.umd.edu/~blj/papers/thesis-PhD-paulr-HMC.pdf>.
- Satyanarayanan, M. Edge computing: A new disruptive force. Keynote address at the 13th Annual ACM Intern. Systems and Storage Conf., 2020.
- Seshadri, V. et al. Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology. In *Proceedings of the 50th Annual IEEE/ACM Intern. Symp. Microarchitecture*, 2017, 73–287; <https://dl.acm.org/doi/10.1145/3123939.3124544>.
- Shafiee, A. et al. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News* 44, 3 (2016), 14–26; <https://dl.acm.org/doi/10.1145/3007787.3001139>.
- Talpes, E. et al. Compute solution for Tesla's full self-driving computer. *IEEE Micro* 40 (2020), 25–35; <https://ieeexplore.ieee.org/document/9007413>.
- Zheng, H., Hu, H., Han, Z. Preserving user privacy for machine learning: Local differential privacy or federated machine learning? *IEEE Intelligent Systems* 35, 44 (2020), 5–14; <https://ieeexplore.ieee.org/document/9144394>.

Gabriel Falcao is a researcher at Instituto de Telecomunicações and a tenured assistant professor at the Department of Electrical and Computer Engineering of the University of Coimbra, Portugal.

João Dinis Ferreira is a research and teaching assistant at ETH Zurich, Switzerland.

Copyright held by owner/author.
Publication rights licensed to ACM.

Early insights and opportunities of AI-powered pair-programming tools.

BY CHRISTIAN BIRD, DENAE FORD, THOMAS ZIMMERMANN, NICOLE FORSGREN, EIRINI KALLIAMVAKOU, TRAVIS LOWDERMILK, AND IDAN GAZIT

Taking Flight with Copilot

IN PAIR PROGRAMMING, two developers write code together. One takes the role of *driver*, writing the code needed for the task at hand, while the other assumes the role of *navigator*, directing the driver's work and reviewing the code. This allows the driver to focus on detailed coding—syntax and structure—while letting the navigator direct the flow of the work and review the code in real time. Proponents of pair programming say it improves code quality and readability and can

speed up the reviewing process.⁵ To date, effective pair programming has required the complex coordination of getting two programmers to work together synchronously. This has made it challenging for teams to adopt this approach at scale. The emergence of new AI-powered tools to support programmers has shifted what it means to pair program.

GitHub Copilot is an AI-powered developer tool leading this shift. GitHub released Copilot in a complimentary technical preview on June 29, 2021, letting hundreds of thousands of developers try coding with an AI pair programmer. Copilot became generally available as a paid product on June 21, 2022. This article focuses on the earliest releases of Copilot—the free technical preview—because these allowed us to capture some of the first experiences with an AI pair programmer. While some changes to Copilot have been made in the version released in 2022, the user interface and experience is largely the same.

With developers taking the role of navigator, they can direct the detailed development work and review the code as it is being written. In addition, the AI assistant can write code (directed by the developer navigator) much faster than a peer, potentially speeding up the process.

Copilot received public attention quickly, generating conversation in forums, press, and social media. These impressions ranged from excitement about potentially boosting developers' productivity to apprehension about AI replacing programmers in their jobs.

But what were developers' actual experiences with Copilot, beyond the hype found in top tweets, *Hacker News*, or Reddit posts? During the early days of the technical preview, we investigated the initial experiences of Copilot users. This provides an unique opportunity to watch how developers would use it, as well as what challenges they encountered. While most of the observations were expected, there were some surprises as well. This article presents highlights from these initial



Table 1. Summary of studies included in this article.

Study 1: Forum Analysis Oct/Nov 2021	Study 2: Case Study Dec/Jan 2022	Study 3: Survey Feb/Mar 2022
What are people using Copilot for?	How are first-timers engaging with Copilot?	How does Copilot impact productivity?
Approach: Reviewed and analyzed 279 GitHub Discussions forum posts	Approach: Conducted a think-aloud study with 5 expert Python developers	Approach: Analysis of 2047 survey responses from Copilot developers
Key Finding: Participants reported spending less time on Stack Overflow, but now have less of an understanding of how or why the code works.	Key Finding: Participants accept the suggestion for efficiency but give up a small bit of autonomy/control over the code they're writing. We observed participants wrestle with this in real-time.	Key Finding: We were able to correlate 11 usage metrics to perceived productivity. Acceptance rate had the highest positive correlation to aggregate perceived productivity.

Example of screenshot of IDE with Copilot code completion.

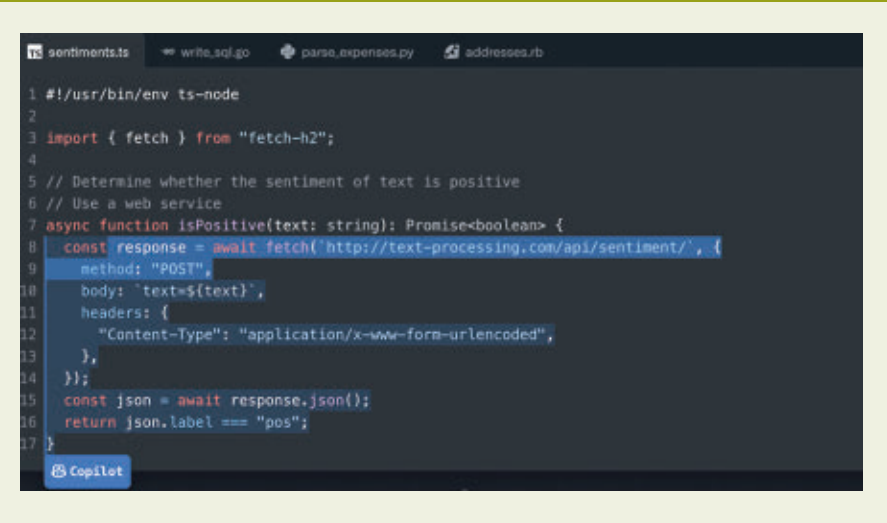


Table 2. Early experiences with Copilot.

The good	The bad	The mixed
users reported productivity improvements	users report Copilot gets into loops of suggesting the same thing	less coding but more reviewing
users compared Copilot to a human	Copilot does not write "defensive code" for example, check null pointers	less time on Stack Overflow, but now less understanding of how/why the code works
"Copilot removes the ceiling on creativity"	Copilot sometimes suggests inappropriate text	API discoverability is supported but does not provide enough information to select best solution
some said Copilot suggested something they would "ordinarily overlook"	Copilot at times leaks PII in header files	

empirical investigations. We should note the models and technology used to develop Copilot are changing rapidly. This analysis was current as of January 2022. Although some features of the tool have evolved, the general themes discussed here still hold true at the date of publication. These highlights include:

- The diverse ways that developers

are using Copilot, including things that worked great and things that very much didn't.

- Challenges developers encountered when using the technical preview of Copilot, yielding insights into how to improve AI pair-programming experiences.
- How AI pair programming shifts the coding experience and reveals the

importance of different skills—for example, the emerging importance for developers to know how to *review* code as much as to *write* code.

- A discussion of opportunities that AI presents to the software development process and its potential impact.

We conducted three studies to understand how developers use Copilot:

1. An analysis of forum discussions from early Copilot users.
2. A case study of Python developers using Copilot for the first time.
3. A large-scale survey of Copilot users to understand its impact on productivity.

We will discuss each of these studies (summarized in Table 1).

What Is Copilot and How Does It Work?

At its core, Copilot consists of a large language model and an integrated development environment (IDE) extension that queries the model for code suggestions and displays them to the user. You may already have interacted with such models when writing documents or text messages. These models have been trained on billions of lines of text and have developed the ability to predict, with high accuracy, what you are going to type next.

The important difference here is that Copilot uses Codex, a language model trained on source code instead of text (for example, email, text messages, websites, or documents). This source code came from a large portion of the public code on GitHub, the largest repository of source code in existence.

According to OpenAI, the team that built Codex, the model has been trained on more than 159GB of Python code alone, in addition to code from many other languages. As such, it's quite common for code that a developer is writing to be like some combination of pieces of code that Copilot has seen before (during model training). Copilot recognizes these similarities and offers suggestions based on the similar pieces of code on which it was trained. Copilot, however, doesn't make suggestions to developers by making copies of code that it has seen previously. Rather, Copilot generates new suggestions (some of which may not actually exist in any code base) by synthesizing what it has

observed in billions of lines of code during model training.

While Copilot leverages this very large model, more than a high-quality code-suggestion engine is required to help developers be more productive. Copilot has been incorporated into multiple IDEs in a way that makes code suggestions timely and seamless. As you write code, requests are continuously sent to Copilot's AI model, which is optimized to analyze the code, identify useful suggestions, and send them back in fractions of a second so that they can be offered to developers in their IDEs when needed, as shown in the accompanying figure.

The experience developers encounter is like the existing autocompletion that has been in modern IDEs for years. The difference is that these suggestions may be longer, sometimes spanning multiple lines of code, and ideally more contextually relevant and helpful. Copilot can currently be used in VS Code, Visual Studio, Neovim, and JetBrains IDEs, including PyCharm and IDEA.

During these initial investigations of Copilot, it became evident that it was unclear to developers how it worked. Among the most common misunderstandings was that while Copilot does learn from code, the learning happens during a general training phase, where OpenAI trains a general-purpose programming model (Codex) that is fine-tuned by using a selected set of public codebases on GitHub. (For more about Codex, a model fine-tuned by OpenAI, refer to <https://openai.com/blog/openai-codex/>)

In contrast, when a developer uses Copilot to generate suggestions during a coding session, the code sent to Copilot to generate the suggestion is not used to make suggestions to other developers. Simply put, this model will not “leak” code to anyone else. In fact, such Web requests are ephemeral, and the code is not logged, captured, or recorded (unless users agree to collection).

Why Are People Using Copilot?

To get a better understanding of how people were using Copilot, we collected and analyzed users' self-reported experiences to uncover the challenges they faced and the opportunities this tool

presented them, as well as to identify how the technology could evolve. The Copilot Discussion forum, available to all technical preview users, was a valuable resource. In this forum, Copilot users provided code snippets, links to blogs, and even live coding videos. An analysis of all 279 posts available, which were devoted to general feedback and personal showcases, uncovered strengths and challenges of Copilot, as well as uses that extend beyond traditional coding tasks.

Among the strengths of Copilot highlighted by users: It helped them adapt to different writing styles, generate simple methods and classes, and even infer which methods to write based on comments. This flexibility is largely a result of the context used by Copilot to make its recommendations. It uses information about the currently active file and code region to prompt the Codex model and adapt the recommendations to developers. Users reported that Copilot can adapt to different coding styles (that is, naming, formatting). In one instance, Copilot was able to make recommendations in someone's own custom programming language. The ability to work with multiple languages was also cited as a strength.

Challenges that Copilot users raised included the risk of revealing secrets such as API keys and passwords or suggesting inappropriate text. This feedback was addressed during the technical preview by adding a filter to Copilot that removed problematic suggestions. Users reported that Copilot sometimes does not write “defensive code”—for example, when Copilot writes a complete method that takes a parameter, it may not check if pointers are null or if array indices are negative. In some cases, users found Copilot suggestions to be distracting and, as such, requested keyboard shortcuts for turning Copilot on/off or muting it for a period. Users also asked that the context should take more than the current file into account.

In addition to traditional AI pair programming (that is, code completion that people think of when using AI in a programming environment), Copilot was used for tasks that extended well beyond this scenario. A few interesting use cases emerged where par-

ticipants used Copilot to help them learn a new programming language, set up a quick server, calculate math problems, and assist in writing unit tests and regular expressions.

These first examples are related to coding but are much more powerful than simple line completion. In fact, one developer passed two coding skills tests without prior knowledge of complex code in the tested languages by using knowledge of other languages and leveraging Copilot for support. Several developers used Copilot to translate text messages from one language to another—for example, from English to French. Another developer connected Copilot with a speech-recognition tool to improve accessibility by building a code-as-you-speak feature. The range of user experiences we observed during the technical preview of Copilot are summarized with examples in Table 2.

Some early users had questions about how Copilot creates its code suggestions. They expressed uncertainty about licensing the code generated by Copilot. Some believed aspects of Copilot, such as code suggestions, might be required to be released under open licenses because the code used to train Codex (the AI model powering Copilot's suggestions) was potentially subject to open source license. Others chimed in with different perspectives, overall expressing different opinions on whether Copilot's suggestions should be released under open licenses or if used by developers, could pose license compliance issues based on inclusion of code suggestions. The rationale for these opinions varied: Some believed that open source licenses applicable to code used to train Copilot somehow applied; others pointed to copyright law; and still others posited a “moral basis” for giving back to open source developers.

There were also discussions about how copyright applied to Copilot's code suggestions. Some users questioned whether code suggested by Copilot would be protected by copyright, and if so, who might assert the copyright—GitHub, a developer using Copilot on a project, or even owners of code used to train Copilot (in cases where Copilot's suggestions matched the code used to train Copilot). The discussions

revealed that many developers may be unfamiliar with global copyright laws. Similarly, the discussions revealed that developers have differing perspectives about whether code generated by use of an AI model should be capable of copyright protection.

Readers should note this section summarizes the comments in Copilot Discussion forum posts and that users did not use precise terms. Also note this summary presents a synopsis of the posts as they were observed and does not represent any personal opinions of the authors or official stance of the authors' employers. Finally, note that while we have endeavored to summarize the comments to our best ability, we are not legal professionals, and this summary is not intended to serve as a legal review of these topics.

How Are First-Time Users Engaging with Copilot?

Findings from the forum analysis inspired several questions about Copilot use in practice. Thus, to better understand how developers engage with this tool in situ, we conducted an in-depth case study with five professional Python developers, strategically selected as external industry developers who had not interacted with Copilot before. The study was conducted according to the following protocol:

1. Participants were walked through a short demonstration of Copilot.
2. After a brief description of the requirements, participants were asked to implement a command-line tic-tac-toe game.
3. Participants were asked to implement a "Send email" feature.
4. A post-study reflective interview was conducted to assess the participants' overall perspective on Copilot.

The order of these tasks was chosen strategically: First to provide guidance to the participants on how to use Copilot; then have them use the tool to independently build their own foundation and mental models of how things work; and finally, to create a scenario where they would likely need to look up an API they do not use often. We hypothesized that recalling the correct use of APIs is where Copilot can provide additional value because it combines code completion, that is, code that developers

would likely be able to provide themselves, as well as additional information that developers might typically need to search for (for example, by searching Stack Overflow). Additional details can be found in the "Study Protocol" sidebar.

At the completion of the case study, researchers convened to discuss participant responses and common themes that emerged. Participants' experiences ranged from enjoyment, guilt, skepticism, clarity on how to interact with the tool, and opportunities for AI pair programming to evolve.

In terms of enjoyment, many participants expressed overall amazement during their first interactions with the tool: *"Wow. That saved me a heck of a lot of time. Yeah, I think I would've tried to do it all in one line and throw some line breaks in there. But this looks better and it's actually a bit more readable."*

Following this first interaction, the light guilt of having this tool at their disposal set in for some participants. One described it as: *"This is pretty cool. At the same time, [Copilot can] make you a little lazy when thinking about the logic you want to implement. You'll just focus on that logic, which it's providing to you, instead of thinking of your own [logic]."*

A core highlight from this case study was the early indicator that developers using AI-assisted tools often spend more time reading and reviewing code than writing, findings supported by other investigations.⁴ The insights from study participants also highlighted some promising areas to investigate in the future, such as including more context around suggestions presented (for example, which suggestion optimizes for readability, performance, or conciseness), as well as code provenance.

Finally, all the participants remarked that they felt Copilot contributed to their productivity. This led to further investigation and the final study covered in this article.

How Does Copilot Impact Productivity?

To better understand how Copilot usage can impact productivity directly, we conducted a survey with users about their perceived productivity and compared it with directly measur-

able usage data. (For a full description and findings of this study, see Ziegler et al.⁷)

Survey participants were those users in the technical preview who opted in to receive communications. Of the 17,420 surveys sent, 2,047 responses were received between February 10, 2022, and March 12, 2022. The survey included questions along several dimensions of productivity based on the SPACE framework.²

Analysis of their responses showed that users' perceived productivity correlated to 11 usage metrics, including persistence rate (percentage of accepted completions that remained unchanged after various intervals); contribution speed (number of characters in accepted completions per hour); acceptance rate (percentage of shown completions accepted by the user); and volume (total number of completions shown to the user). The paper includes a full list of the adapted metrics used. Across all analyses, acceptance rate has the highest positive correlation to aggregate perceived productivity ($\rho = 0.24$, $P < 0.0001$)—higher than persistence measures.⁷ This finding implies that users perceive themselves to be more productive when accepting suggestions from Copilot, even if they must edit the suggestion. That said, this presents opportunities to explore how suggestions that help users think and tinker may be more valuable to them than ones that save them typing time.⁶

Discussion

Copilot is one of the first widely used developer tools powered by AI models, offering a notable shift over previous methods like genetic programming.³ Over the next five years, however, AI-powered tools likely will be helping developers in many diverse tasks. For example, such models may be used to improve code review, directing reviewers to parts of a change where review is most needed or even directly providing feedback on changes. Models such as Codex may suggest fixes for defects in code, build failures, or failing tests. These models can *write* tests automatically, helping to improve code quality and downstream reliability of distributed systems.

AI models could help refactor large, complex code bases, making them easier to read and maintain. Code comments or documentation may be automatically generated using these models. In short, any developer task that requires interacting or reasoning about code in any way can likely be aided by AI. The challenge will come in creating the right user experience such that the developer is helped more than hindered.

This study of Copilot shows that developers spend more time reviewing code (as suggested from Copilot or similar tools) than writing code. As AI-powered tools are integrated into more software development tasks, developer roles will shift so more time is spent assessing suggestions related to the task than doing the task itself (for example, instead of fixing a bug directly, a developer will assess recommendations of bug fixes).

In the context of Copilot, there is a shift from *writing* code to *understanding* code. An underlying assumption is that this way of working—looking over recommendations from tools—is more efficient than doing the task directly without help. These initial user studies indicate that this is true, but this assumption may not always hold for varying contexts and tasks. Finding ways to help developers understand and assess code—and the context within which that code executes and interacts—will be important. This naturally leads to the need to understand the dynamics between developers and AI-powered tools.

An increasingly important topic of consideration is the trust that developers have in AI-powered tools like Copilot. The success of any new development tool is tied to how much a developer trusts the tool is doing the “right thing.” What factors are developers finding important to build that trust? How does trust get reconstructed after AI-powered developer tools perform in an unexpected manner? For example, if code suggested by Copilot introduces security vulnerabilities or performance bottlenecks, its use will rapidly decline.

“Traditional” tools such as compilers or version control systems are largely deterministic and predictable. When problems occur, developers can examine and even modify the source

Study Protocol

The Copilot case study took place over two days. The team spent an hour with each participant. Each interview took place over Microsoft Teams. The researchers used a PowerPoint presentation to organize the study, and participants were asked to share their screens as they engaged with Copilot. Each interview consisted of the following steps:

Task 1: Small Demonstration of Copilot

Participants were asked to launch Copilot via a preconfigured Codespace linked to a GitHub repository. From there participants were asked to complete a basic function that accepts an integer and returns whether that integer is a prime number. As discoverability was not part of the study, descriptions and guidance were provided, pointing out features of Copilot as they presented themselves.

Task 2: Creating a Command-line Tic-Tac-Toe Game

Once participants understood how to engage Copilot and its basic functionality, they were asked to build a tic-tac-toe game with the following criteria:

1. Add a class for a tic-tac-toe board that defines the nine cells of the game board.
2. Add a method that accepts and tracks player moves on the board.
3. Add a method that displays the board in the command line.
4. Add a method that determines when the game is over and whether a player has won.
5. Write code to test that the board is displayed correctly and that the game is over when a player wins.

Task 3: Implement a “Send Email” Feature

To get a sense of how participants would respond to using an unfamiliar API (and how Copilot would respond with suggestions), they were asked to build a feature to email the researcher when a game is complete. All participants noted that they had never used the `smtplib` library before.

Post-Study Reflective Interview

After roughly 35–40 minutes of using Copilot to complete these tasks, participants were asked a set of rating-scale questions, concluding with one open-ended question:

1. How would you rate your overall experience with Copilot today?
(1 = “Not at all positive” to 5 = “Very positive”)
2. How helpful was Copilot while you attempted to complete the tasks today?
(1 = “Not at all helpful” to 5 = “Very helpful”)
3. How effective were Copilot’s suggestions while attempting to complete the tasks today? (1 = “Not at all effective” to 5 = “Very effective”)
4. How different were Copilot’s suggestions from your typical “style” of writing code? (1 = “Not at all different” to 5 = “Very different”)
5. How interruptive was Copilot while you were trying to complete the tasks today? (1 = “Very interruptive” to 5 = “Not at all interruptive”)
6. How likely would you be to recommend Copilot to a friend or colleague? (1 = “Not at all likely” to 5 = “Very likely”)
7. How likely would you be to use Copilot if it was available today? (1 = “Not at all likely” to 5 = “Very likely”)
8. How disappointed would you be if GitHub decided to discontinue Copilot? (1 = “Not at all disappointed” to 5 = “Very disappointed”)
9. How would you describe Copilot to one of your colleagues who has not heard of it before?

code to understand any unexpected behavior. That is simply not the case with AI-powered tools. In fact, AI deep-learning models are probabilistic and more opaque. Further research is needed to better understand how tools leveraging these AI models can be designed to foster developer trust, leading to a measurable positive impact with developers.

Finally, it will become important to track AI-generated code throughout the software development life cycle, as this will help answer important questions: Does AI-generated code lead to fewer

(or more?) build breaks, test failures, or even post-release defects? Should such code have scrutiny during code review? What proportion of shipping code comes from tools such as Copilot?

The answers to these questions are important to all stakeholders of a software organization but answering them requires knowing where each line of code comes from. Unfortunately, these answers are unknown right now: The provenance of generated code doesn’t live past a single development session in an IDE. There is no

Research Pointers Hot Off the Press

AI pair programming is a relatively new area that practitioners and researchers are actively exploring. If you are interested in reading more about the topic, there are some recent papers and studies, organized in two main categories: How programmers are using Copilot; and the effectiveness of Copilot. (Note that Copilot was the first AI pair programmer to be released and was the tool with the largest distribution, so most research is done with this tool.)

How Programmers are Using Copilot

- ▶ Vaithilingam, P., Zhang, T., Glassman, E. Expectation vs. experience: evaluating the usability of code generation tools powered by large language models. *Extended Abstracts of the 2022 Conf. Human Factors in Computing Systems*, Article no. 332, 1–7; <https://dl.acm.org/doi/10.1145/3491101.3519665>;
- ▶ Barke, S., James, M. B., Polikarpova, N. *Grounded Copilot: how programmers interact with code-generating models*. 2022; <https://arxiv.org/abs/2206.15000>.

Vaithilingam et al. share insights on how 24 students use Copilot for three programming tasks and its impact on task completion time and success rate. Barke et al. report on a grounded theory analysis of how 20 programmers interacted with Copilot. They observed modes: In acceleration mode, a programmer uses Copilot to get to the code faster; in exploration mode, a programmer uses Copilot to explore the options.

Effectiveness of Copilot

- ▶ Dakhel, A.M., Majdinasab, V., Nikanjam, A., Khomh, F., Desmarais, M.C., Jiang, Z.M. GitHub Copilot AI pair programmer: Asset or liability? 2022; <https://arxiv.org/abs/2206.15331>.
- ▶ Nguyen, N., Nadi, S. An empirical evaluation of GitHub Copilot's code suggestions. In *Proceedings of the IEEE/ACM 19th Intern. Conf. Mining Software Repositories*, 2022, 1–5; <https://dl.acm.org/doi/10.1145/3524842.3528470>.
- ▶ Imai, S. Is GitHub Copilot a substitute for human pair-programming? An empirical study. In *Proceedings of the IEEE/ACM 44th Intern. Conf. Software Engineering: ICSE-Companion*, 319–321; <https://dl.acm.org/doi/abs/10.1145/3510454.3522684>.
- ▶ Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B., Karri, R. Asleep at the keyboard? Assessing the security of GitHub Copilot's code contributions. In *Proceedings of the IEEE 2022 Symp. Security and Privacy*, 754–768; <http://bit.ly/40GbMTQ>.
- ▶ Asare, O., Nagappan, M., Asokan, N. 2022. *Is GitHub's Copilot as bad as humans at introducing vulnerabilities in code?* Arxiv.org; <https://arxiv.org/abs/2204.04741>.
- ▶ Ziegler, A., Kalliamvakou, E., Li, X. A., Rice, A., Rifkin, D., Simister, S., Sittampalan, G., Aftandilian, E. Productivity assessment of neural code completion. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, 2022, 21–29; <https://dl.acm.org/doi/10.1145/3520312.3534864>.

Dakhel et al. compare Copilot's solution for fundamental algorithmic problems with human-written code by students. Nguyen and Nadi investigate how well Copilot performs on LeetCode questions and compare its performance across programming languages. Imai investigates the effectiveness of Copilot as a substitute for a human pair programmer. Several papers focus on security. Pearce, et al. found that just like human developers, Copilot can produce vulnerable code in some cases. Asare et al. are currently working on an empirical study to investigate whether Copilot is as likely as human developers to introduce vulnerabilities. Ziegler et al. investigate whether developer interactions with GitHub Copilot can predict self-reported productivity and report patterns in the acceptance rates of Copilot suggestions. (A portion of this work is briefly summarized in the article's discussion about the large-scale survey.)

way to know which code checked into a git repository comes from an AI tool. Developing provenance tools that can track generated code end to end from IDE to deployment will be critical for software organizations to make informed decisions about when, where, and how to incorporate AI-powered tools into their development.

Conclusion

This research on Copilot provided early insight into how AI-powered tools are making an entrance into the software-development process. Likewise, these studies have also presented new research questions that warrant further investigations for AI-powered tools overall.¹ We hope these early

findings inspire readers to consider what this can mean for the nature of collaboration for their work in the future and its potential impact. The sidebar “Research Pointers Hot Off the Press” lists some recent work on AI-powered programming.

Acknowledgments. We thank the Copilot team for great discussions and our study participants for offering great insights. 

References

1. Ernst, N.A., Bavota, G. AI-driven development is here: Should you worry? *IEEE Softw.*, 39, 2, (2022), 106–110; <https://ieeexplore.ieee.org/document/9713901>.
2. Forsgren, N., Storey, M.A., Maddila, C., Zimmermann, T., Houck, B., Butler, J. The SPACE of developer productivity: There's more to it than you think. *queue* 19, 1 (2021), 20–48; <https://queue.acm.org/detail.cfm?id=3454124>.
3. Sobania, D., Briesch, M., Rothlauf, F. Choose your programming copilot: a comparison of the program synthesis performance of GitHub Copilot and genetic programming. In *Proceedings of the 2022 Genetic and Evolutionary Computation Conference*, 1019–1027; <https://dl.acm.org/doi/10.1145/3512290.3528700>.
4. Vaithilingam, P., Tianyi, Z., Glassman, E. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Proceedings of the 2022 Conf. Human Factors in Computing Systems Ext. Abstracts*, 1–7; <https://doi.org/10.1145/3491101.3519665>.
5. Williams, L. Pair programming. *Making Software: What Really Works, and Why We Believe It*. A. Oram and G. Wilson, eds. O'Reilly Media, 2011, 311–328.
6. Ziegler, A. Research: How GitHub Copilot helps improve developer productivity. *GitHub Blog*, 2022; <http://bit.ly/3I9tUye>.
7. Ziegler, A. et al. Productivity assessment of neural code completion. In *Proceedings of the 6th ACM SIGPLAN Intern. Symp. Machine Programming* 2022, 21–29; <https://doi.org/10.1145/3520312.3534864>.

Christian Bird is a senior principal researcher in the Software Analysis and Intelligence (SAINTES) group at Microsoft Research. His work has focused on code review, branching and merging, developer productivity, and applying AI/ML to software engineering tasks.

Denae Ford is a senior researcher at Microsoft Research in the SAINTES group and an affiliate assistant professor in the Human Centered Design and Engineering Department at the University of Washington, Seattle, WA, USA.

Thomas Zimmermann is a senior principal researcher in the Software Analysis and Intelligence (SAINTES) group at Microsoft Research. He is best known for his work on software analytics and data science in software engineering.

Nicole Forsgren is a partner at Microsoft Research, where she leads Developer Velocity Lab. Her current work investigates AI and its role in transforming the software engineering process.

Eirini Kalliamvakou is a staff researcher at GitHub Next, where she leads user studies that shape the team's understanding of user needs and guide the iteration of prototypes.

Travis Lowdermilk is a principal UX researcher in the Developer Division at Microsoft. His work focuses on connecting product teams with their customers to uncover unmet needs and build innovative products.

Idan Gazit is a senior director of research at GitHub Next, where he leads the Developer Experiences research team. Prior to that, he led the Heroku Data UX team, and is an alumnus of the Django Web Framework core team.

Copyright held by owner/author.
Publication rights licensed to ACM.

OPEN FOR SUBMISSIONS

ACM Journal on Computing and Sustainable Societies (JCSS)

Editor-in-Chief

Lakshminarayanan Subramanian
New York University, USA



Publishing significant and original interdisciplinary research from a broad array of fields that support the growth of sustainable societies worldwide

Inspired by the broad agenda of the United Nations Sustainable Development Goals (SDGs), the *ACM Journal on Computing and Sustainable Societies* (JCSS) aims to publish significant and original research from a broad array of computer and information sciences, social sciences, environmental sciences, and engineering fields that support the growth of sustainable societies worldwide, especially including under-represented and marginalized communities. JCSS aims to explicitly promote interdisciplinary research work including new methodologies, systems, techniques, applications, behavioral, qualitative, and quantitative studies that addresses key societal challenges including sustainability, gender equality, health, education, poverty, accessibility, conservation, climate change, energy, infrastructure and economic growth, among others. We also welcome research on the ethics of technology, especially from a critical perspective, that explores limitations and concerns with technology-led solutions for sustainable societies. The journal will be published quarterly.

For more information and to submit your manuscript, please visit acmjcss.acm.org.



Association for
Computing Machinery

DOI:10.1145/3560469

Today's powerful, robust SAT solvers have become primary tools for solving hard computational problems.

BY JOHANNES K. FICHTE, DANIEL LE BERRE, MARKUS HECHER, AND STEFAN SZEIDER

The Silent (R)evolution of SAT

THE PROPOSITIONAL SATISFIABILITY problem (SAT) was the first to be shown NP-complete by Cook and Levin. SAT remained the embodiment of theoretical worst-case hardness. However, in stark contrast to its theoretical hardness, SAT has emerged as a central target problem for efficiently solving a wide variety of computational problems. SAT solving technology has continuously advanced since a breakthrough around the millennium, which catapulted practical SAT solving ahead by orders of magnitudes. Today, the many flavors of SAT technology can be found in all areas of technological innovation.

SAT asks whether a given propositional formula is *satisfiable*. That is, can we set the formula's variables to values 1 (True) or 0 (False) in such a way that the entire formula evaluates to 1? $F = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2) \wedge (x_2 \vee x_3)$ is a simple propositional formula in conjunctive normal form (CNF), where x_1 , x_2 , and x_3 are propositional variables

and $\vee$, $\wedge$, and $\neg$ refer to the logical operators OR (disjunction), AND (conjunction), and NOT (negation), respectively. A variable x_i or a negated variable $\neg x_i$ is a *literal*, and a disjunction of literals is a clause. So, the above formula F is a conjunction of four clauses. The formula is satisfiable; we can satisfy it by the truth assignment that sets x_1 and x_2 to 1, and x_3 to 0: the first, third, and fourth clauses are satisfied by $x_2 = 1$ because the clauses contain x_2 . The second clause is satisfied by $x_3 = 0$ because it contains $\neg x_3$. In consequence, all clauses are satisfied. A truth assignment naturally extends from variables to literals by setting $\neg x$ to the opposite value of x . Hence, a formula is satisfiable if and only if there is a truth assignment that sets at least one literal in each clause to 1.

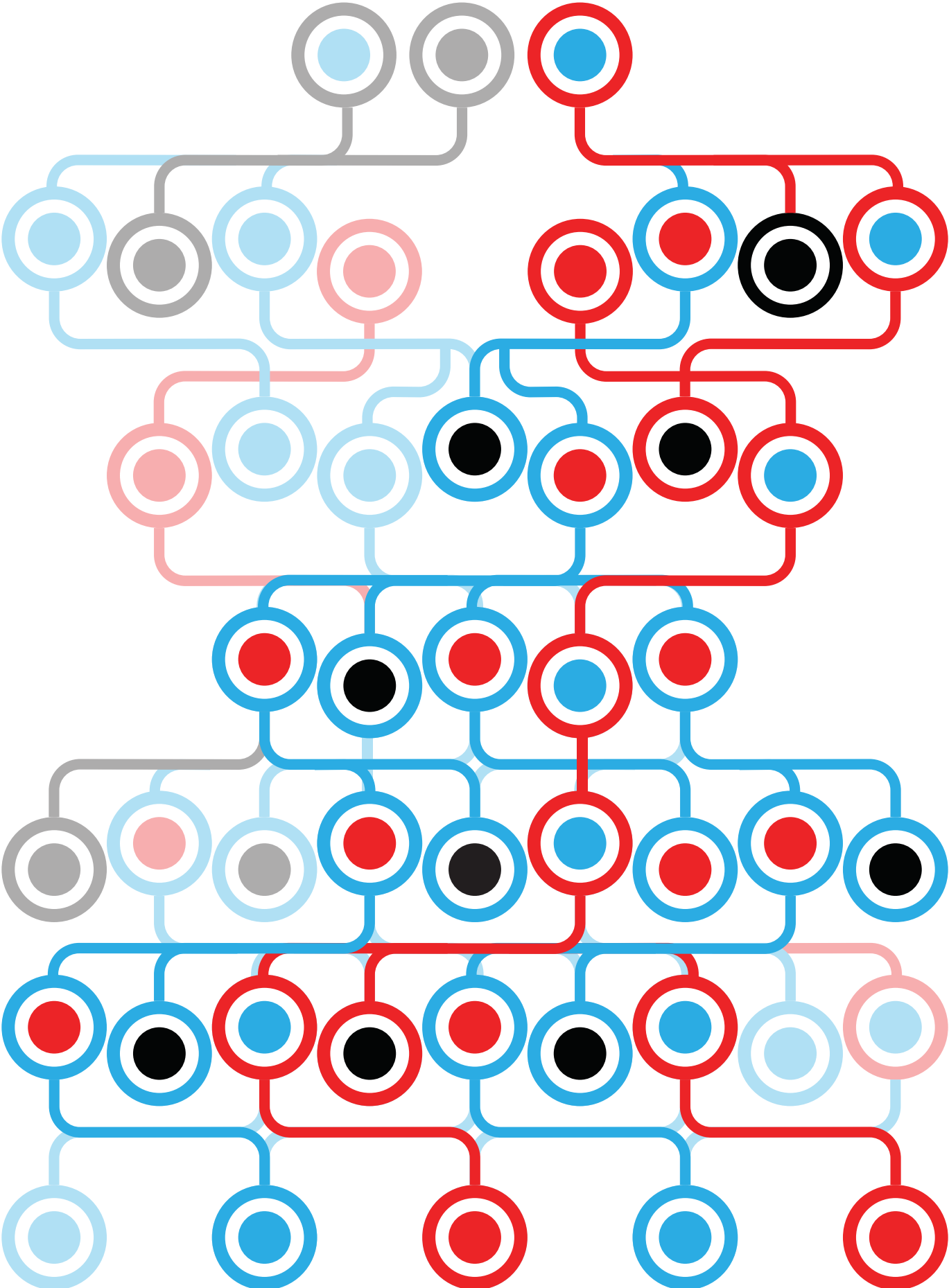
Example 1 shows a larger formula that is *unsatisfiable*—that is, not satisfied by any assignment. The focus on CNF formulas is not a restriction. The so-called Tseitin transformation³⁹ efficiently transforms any propositional formula into CNF without affecting its satisfiability.

At first glance, the SAT problem looks inconspicuous since it is simple to state, does not look difficult to solve, and seems uninteresting for practical purposes. Still, Stephen Cook⁷ and Leonid Levin²⁹ showed independently in the 1970s that SAT is NP-complete, making it the first NP-complete problem. So, suppose one could solve SAT in polynomial time on arbitrary input. In

>> key insights

- **Propositional Satisfiability (SAT) has been a cornerstone of computational complexity theory; now, it has become a central target problem for solving hard computational problems in practice.**
- **Since the revolution in SAT solving for decision problems that took place around the millennium, significant efficiency improvements have been achieved, and new methods for certification and trust have been added.**
- **Over the last 10 years, SAT further evolved by broadening its applications, including optimization, counting, and even problems involving quantifiers.**

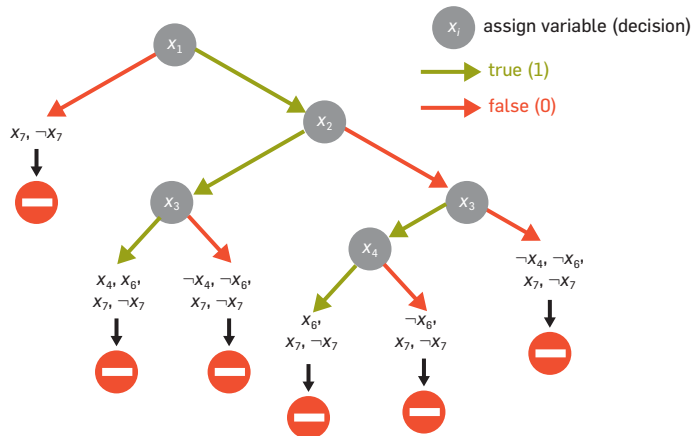
IMAGE BY IGOR KISSELEV



Example 1. Under each of the 2^9 truth assignments to the variables $x_1, \dots, x_9$, at least one of G 's clauses evaluates to 0, making the formula G unsatisfiable.*

$$\begin{aligned}
 G = & \overbrace{(x_1 \vee \neg x_2)}^{c_1} \wedge \overbrace{(x_1 \vee \neg x_3)}^{c_2} \wedge \overbrace{(x_1 \vee \neg x_8)}^{c_3} \wedge \overbrace{(x_1 \vee \neg x_7)}^{c_4} \wedge \\
 & \overbrace{(\neg x_4 \vee x_3)}^{c_5} \wedge \overbrace{(\neg x_5 \vee \neg x_6)}^{c_6} \wedge \overbrace{(\neg x_2 \vee \neg x_3 \vee x_4)}^{c_7} \wedge \\
 & \overbrace{(\neg x_4 \vee x_6 \vee x_9)}^{c_8} \wedge \overbrace{(\neg x_3 \vee \neg x_7 \vee x_8)}^{c_9} \wedge \overbrace{(x_1 \vee x_8)}^{c_{10}} \wedge \overbrace{(x_1 \vee x_7)}^{c_{11}} \wedge \\
 & \overbrace{(x_4 \vee x_5 \vee \neg x_1)}^{c_{12}} \wedge \overbrace{(x_4 \vee \neg x_6 \vee \neg x_1)}^{c_{13}} \wedge \overbrace{(\neg x_6 \vee \neg x_4 \vee \neg x_1)}^{c_{14}} \wedge \\
 & \overbrace{(x_6 \vee \neg x_7 \vee \neg x_1)}^{c_{15}} \wedge \overbrace{(\neg x_7 \vee x_6 \vee \neg x_1)}^{c_{16}} \wedge \overbrace{(x_7 \vee x_6 \vee \neg x_1)}^{c_{17}}
 \end{aligned}$$

Example 2. Search Tree: Illustration of a possible run of the DPLL algorithm on the formula G from Example 1. Circles indicate decision variables, and unit propagation is represented by the list of propagated literals. After obtaining inconsistency ($\perp$) in one branch, DPLL chronologically backtracks to the last decision, which in our example causes the search to run into the same conflict several times.



that case, one could also solve any NP-complete problem in polynomial time, and it would follow that P equals NP. Thus, in terms of worst-case complexity theory, SAT embodies computational hardness. Also, in modern complexity theory, SAT continues to serve as a hard benchmark problem in the form of the (Strong) Exponential Time Hypothesis.⁴

In stark contrast to its theoretical worst-case hardness, the SAT problem has emerged as an essential instrument for efficiently solving a wide variety of computational problems, ranging from hardware and software verification to planning, combinatorial design, and software dependency.^{1,4} In this way, SAT

significantly impacts today’s technological innovation. SAT is widely applied in knowledge representation, reasoning,¹⁹ and artificial intelligence (AI).⁹ Although SAT is mainly associated with symbolic AI, it contributes to non-symbolic AI by providing model-counting algorithms, which are essential tools for probabilistic reasoning,⁹ and allocates a main backbone for neurosymbolic AI.²⁴ Notably, using SAT, long-standing open problems in mathematical combinatorics were successfully solved—for example, the Pythagorean Triples Problem.^{4,21} Solving such challenging problems with SAT requires the ability to efficiently translate the original problem into an in-

stance of the SAT problem and the availability of computer programs, called *SAT solvers*, which efficiently evaluate SAT instances. Initial progress in practical SAT solving began in the early 1990s, leading to a breakthrough around the millennium. The last two decades have brought further enormous technological advancement and innovation to SAT solving. Today, SAT solvers are so powerful and robust that they have become primary tools for solving hard computational problems. Solvers have been embedded into complex procedures to solve more complex problems, such as optimization problems (MaxSAT, Pseudo-Boolean Optimization) or quantified satisfiability (QSAT).

“The story of satisfiability is the tale of a triumph of software engineering, blended with rich doses of beautiful mathematics,” writes Donald E. Knuth in the preface to the second part of the fourth volume of *The Art of Computer Programming*,²⁸ which contains a section on satisfiability that stretches well over 300 pages.

As we observed incredible advances in computer hardware, yielding ever-faster processors, large and efficient memory, and massively parallel computing units, one could ask whether progress in SAT solving is the sole result of hardware advancement. A recent *Time Leap Challenge* addressed this question by running a race between 20-year-old SAT solvers on new computer hardware and modern SAT solvers on 20-year-old computer hardware.¹⁶ The experiments confirm Knuth's statement on engineering and mathematics. Although hardware improvements make old solvers faster, algorithmic progress dominates and drives today's SAT solving.

This article aims to shed light on the continuing progress of practical SAT solving through a series of evolutionary innovations and improvements that have been ongoing since the revolutionary breakthrough around the millennium. Overall, we argue that SAT has earned the title of a *silent (r)evolution*. We tell the story of SAT divided into three eras: the pre-revolution, the revolution, and the evolution.

Eras of Practical SAT Solving

Era I: The pre-revolution. The building blocks of today's SAT success are various and date back even to the first half

of the 20th century. We refer to other sources for a detailed description of SAT's history⁴ and focus on a few important milestones of the modern era. Complete and incomplete solvers were the prevalent SAT solvers in the 1990s. *Incomplete solvers*, based on stochastic local search,²³ were successfully applied to planning problems⁴ and satisfiable instances. In contrast, *complete solvers*, based on backtracking search, were used to solve combinatorial problems (such as puzzles, N-queens, and Latin squares) as well as uniform random k-SAT formulas (including unsatisfiable ones). These early complete solvers followed the general approach, called *DPLL*, which was first proposed by Davis, Logemann, and Loveland (DLL) as a memory-efficient refinement of an earlier algorithm by Davis and Putnam (DP).^{8,11,12}

In modern terminology, DPLL is a backtracking algorithm that performs a depth-first exploration of a binary search tree on truth assignments (see Example 2). It applies the following two optimization steps repeatedly after a variable has been assigned, propagating the current partial assignment: if the current assignment sets all but one literal of a clause to 0 (the clause is a unit clause), then we can safely set the remaining unassigned literal to 1—*unit propagation* is the repeated application of this process; if the current assignment satisfies all clauses containing some unassigned literal, then we can safely set that literal to 0—the opposite literal is called a *pure literal*. Backtracking occurs when the current partial assignment sets all the literals of a clause to 0.

Heuristic methods that decide when and how variables are assigned play a crucial role in the context of DPLL.⁸

Early implementation challenges. Three SAT competitions were organized in the 1990s. The second competition, 1993's DIMACS implementation challenge,⁴ introduced the standard ASCII input format for SAT solvers, which is still in use. This standardized input format supports reusing SAT solvers as black boxes and sharing benchmarks. The DIMACS CNF format describes a propositional formula. The preamble provides the format (CNF), the number

of variables, and the number of clauses. The remaining lines denote clauses, where each literal is represented by a signed integer, using 0 as a separator. Example 3 illustrates our running example containing nine variables and 17 clauses in the DIMACS CNF format.

Era II: The revolution. The first pillar of the revolution³⁰ was the solver GRASP,³¹ which proposed a new architecture, combining non-chronological backtracking, conflict analysis, and learning, today referred to as *conflict driven clause learning (CDCL)*. CDCL is more than just DPLL with learning, since it is no longer a pure backtracking search. It captures unit propagation in a directed acyclic implication graph that is used to perform conflict analysis and clause learning, which prominently drives the search. Modern SAT solvers use the *trail* data structure proposed by MiniSat¹³ to capture the search tree and the implication graph.

The second pillar was the CDCL-based solver Chaff,³² specially designed to solve large benchmark instances by taking the characteristics of the host computer into account and achieving an unprecedented balance between the sophistication of algorithms and data structures on the one side and the practical efficiency on the other. Chaff introduced the Watched Literal data structure, a “lazy” scheme for performing unit propagation that allowed cost-free backtracking. The branching heuristics (VSIDS) and conflict analysis procedure were carefully designed with a new tradeoff between reasoning power and computation cost: The emphasis on recent conflicts leads to a locality-based search.

We illustrate CDCL for our running example in Example 4. Many conflict clauses are learned during the search, which is quite demanding on memory. Modern CDCL solvers frequently delete learned clauses and heuristically predict the ones to keep.⁴ Various heuristics, such as VSIDS,³² EVSIDS,¹³ or LRB,⁴ are available for optimizing the search and assigning the variables. Those heuristics impose low overhead, generate almost no cost for backtracking, and are updated when learning clauses. To escape unlucky search tree exploration during the search, solvers frequently unassign all variables and restart the search (but keep the learned

clauses). Common heuristic schemes that decide when to restart are Luby or Rapid Restarts.⁴ Also, machine-learning techniques have been used to optimize heuristics.⁴ Still, more careful analysis is needed to better understand how the components of the solver and their interaction affect its overall efficiency.^{14,27}

Some critical applications drew much attention to practical SAT solving from academia and industry. For example, bounded model checking (BMC)⁴ provided numerous challenging benchmarks, which were soon tackled. BMC, which checks the correctness of sequential systems over a bounded number of steps, was proposed after the failure of standard model checking with binary decision diagrams (BDDs), a graph-based canonical representation of propositional formulas.⁶ The BMC benchmarks were extensive, with tens of thousands of variables and clauses. Combining conflict-driven heuristics with conflict analysis (CDCL) could solve BMC benchmarks up to two orders of magnitude faster than any other approach. These impressive results applied to other application benchmarks as well. The computer-aided verification (CAV) community acknowledged the central role of GRASP and Chaff for that outstanding progress by presenting their authors with the CAV 2009 award for “fundamental contributions to the development of high-performance Boolean satisfiability solvers.”

Era III: The evolution. While improving solvers is an important goal, the community has progressed in new modeling techniques and methods for certifying results.

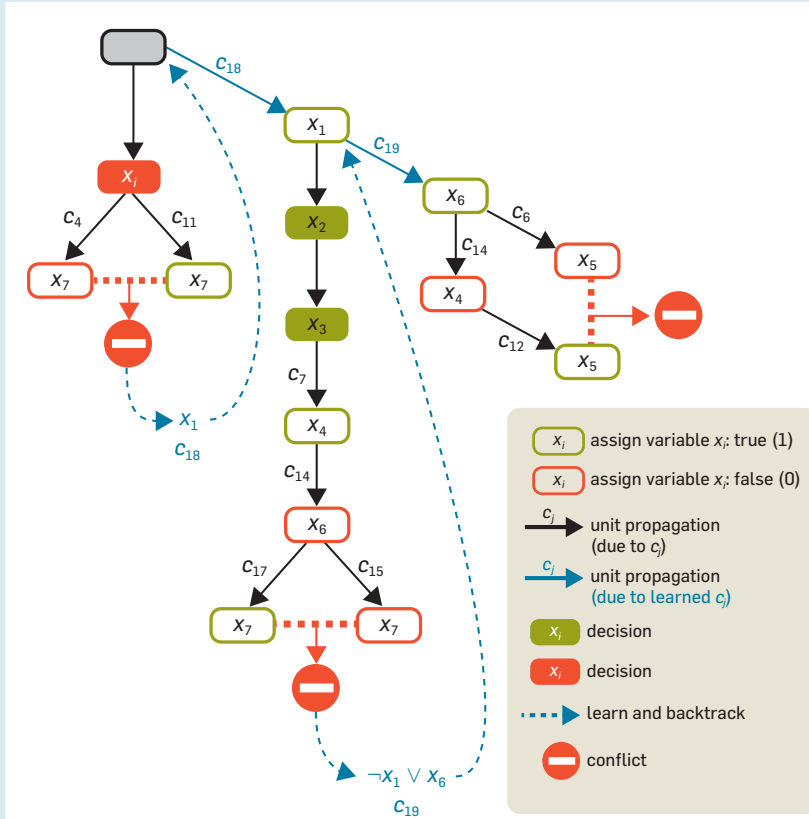
Efficient SAT encodings. It is challenging to express a computational problem as a CNF formula that the solver can solve efficiently. Encodings may blow up the resulting CNF formula quite notably, resulting in quadratic, cubic, or even

Example 3. DIMACS CNF representation of Example 1's formula G. Try this example at <http://bit.ly/406Lqc7> using for instance Kissat SAT solver <http://bit.ly/30a8zx9> to solve it with a real SAT solver.

```
p cnf 9 17 -5 -6 0 4 5-1 0
1 -2 0 -2 -3 4 0 4 -6-1 0
1 -3 0 -4 6 9 0 -6 -4-1 0
1 -8 0 -3 -7 8 0 6 -7-1 0
1 -7 0 1 8 0 -7 6-1 0
-4 3 0 1 7 0 7 6-1 0
```

a DPLL combines DP and DLL to emphasize both contributions.

Example 4. Propagation Graph: Trace of a possible CDCL run on the formula G from Example 1.



Boxes represent assigning a truth value to a variable. Filled boxes represent decisions. Edges mark the underlying reason, which can be a decision (no label) or unit propagation (clause ID label). First, we decide that x_1 is set to 0. By conflict analysis of the clauses causing the conflict (c_4 and c_{11}), we immediately learn the conflict clause $c_{18} = x_1$. Then, we backtrack and after unit propagation, we perform two decisions on variables x_2 and x_3 , which allows us to finally learn the conflict clause $c_{19} = \neg x_1 \vee x_6$. This additional clause, together with unit propagation, already results in the conclusion that G is unsatisfiable, and there are no decisions left for backtracking. In contrast to Example 2, CDCL applies non-chronological backtracking and learning to avoid rediscovering the same conflict several times, by learning—for instance—that when x_1 is set to 1, x_6 has to be set to 1 as well (clause c_{19}). We observe that it allowed us to reduce the number of decisions from five to three, and the number of conflicts from six to three.

Example 5. Encoding techniques based on binary adders.³⁷

Let H be a formula that contains many variables, say $v_1, v_2, \dots, v_{1000}$. Assume that for modeling purposes, we are interested in restricting the formula in such a way that at most one of these variables is allowed to be set to 1. We can achieve this by adding all the $\binom{1000}{2} = 499,500$ binary clauses $\neg v_i \vee \neg v_j$, $1 \leq i < j \leq 1,000$; this generates a huge formula. However, we can avoid this quadratic space requirement and express the restriction much more succinctly with only a linear overhead. Namely, we can mimic well-known logic circuits that form binary adders with carryover bits in terms of clauses, involving the original variables v_i and additional auxiliary variables.

worse overhead compared to the original problem instance. Usually, one faces a tradeoff between space and simplicity. Intuitively, one would try to keep the instance as small and concise as possible, balancing the number of variables and the number of clauses. If we have n variables and add m auxiliary variables, the potential search space grows from 2^n to 2^{n+m} . Still, in practice, adding such variables can reduce the number of clauses, thereby speeding up the search.⁴ A similar pattern was observed by early automatic test pattern-generation programs, such as path-oriented decision making (PODEM).⁴ This is particularly relevant when encoding properties that constrain the number of variables from a set

of variables that must be set to 1, called *cardinality constraints*. We give a simple example in Example 5.

Assumptions. Using SAT solvers as “oracles” is far more complex in practice than just having an NP oracle in theoretical considerations. On the one hand, as successful as SAT solvers are, they are not NP oracles. SAT solvers need time for solving and may not even answer within a reasonable time. On the other hand, modern SAT solvers are more sophisticated than NP oracles. They admit *stateful* procedures, where clauses may be added or removed during solving. To control added clauses without invalidating learned clauses, modern solvers support *assumptions*, popularized by MiniSat,¹³ which are additional literals introduced for representing the state of clauses. It turned out that assumptions are highly versatile and useful. We demonstrate assumptions in Example 6.

Inprocessing/preprocessing. SAT solvers run various simplification rules to eliminate unnecessary variables or clauses. Several simplification rules exist; some can be carried out in linear time while others require quadratic time. Extremely beneficial is preprocessing, where the instance is simplified before the actual solver is started.⁴ Interestingly, one such preprocessing task is to run a restricted form of the Davis-Putnam procedure on a subset of variables (bounded variable elimination). However, modern solvers such as Lingeling implement more sophisticated rules, which are run during the search before the next variables are assigned (inprocessing) and take learned clauses into account.⁴ More time-consuming inprocessing techniques such as equivalence learning, cardinality constraints, or parity detection can be applied with a limited budget and interleaved with search.

Parallel solving. In the 2000s, Moore’s law started approaching its limits in CPU performance on single cores, and the clock-speed improvements for silicon-based chips started to slow down. Parallel computation was one way to compensate for this trend, and it found its way into consumer hardware. Around the same time, parallel computing also started to become more fashionable in SAT solving, aiming to improve overall processing performance and tackle huge instances.²¹ The two main techniques

that evolved are *parallel-portfolio* and *divide-and-conquer* solving.

In parallel-portfolio solving, a problem instance is independently given to a collection of solvers competing for a solution in parallel. Usually, the collection comprises different solvers or one solver with different heuristics. While parallel SAT solvers optimize performance to solve large instances, they may produce different satisfying assignments or unsatisfiability proofs for a given input formula because they are inherently non-deterministic due to race conditions. Nevertheless, some recent parallel solvers achieve determinism with a reasonable runtime overhead. Determinism is critical for applications requiring reproducibility, such as scheduling or model checking. Over the last years, massive parallelization using graphics processing units (GPUs) or tensor processing units (TPUs) has become a popular approach in computer science and ML. Aspects of modern SAT solving have been successfully implemented and tested on GPUs, including solvers tailored for counting the number of satisfying assignments.¹⁷ Still, SAT solving has not yet benefited from massive parallelization.

Divide-and-conquer divides a formula into multiple formulas. The workload is shared among multiple solvers that run in parallel and can even synchronize learned clauses. However, modern sequential SAT solvers do not need to exhaust the search space to find a solution or refute a formula; learned clauses allow for shortcuts, making it hard to decompose instances because of the increased synchronization efforts for learned clauses. A popular version is *cube-and-conquer*, which partitions an instance by a look-ahead technique into up to a million sub-instances and then solves the sub-instances with a CDCL-based solver.²¹

Correctness of solvers and emitting proofs. The results of early SAT solvers were hard to validate. While it is easy to check whether an assignment returned by a solver satisfies the given formula, in the past, no technique was available to verify correctness if the solver claimed that an instance is unsatisfiable. During the first competitions, one could only check whether emitted assignments were correct and that solvers did not provide inconsistent results. If solver A

outputs “satisfiable” and solver B outputs “unsatisfiable” on the same formula, one would check whether the assignment given by A indeed satisfies the formula. If so, solver B is incorrect; otherwise, solver A is incorrect, but nothing can be concluded about solver B. Solver correctness has been addressed by intensive testing, including fuzzing and asking for traces that give rise to an unsatisfiability proof within a dedicated propositional proof system.⁴ Such traces can be automatically checked using third-party tools, which can be specified, implemented, and fully verified. The relationship between CDCL runs and propositional proof systems is now well understood. All the relevant

proof systems rely on the *resolution rule*, which derives a new clause from two clauses containing exactly one opposite literal. For instance, in Example 4, the clause $c_{18} = x_1$ can be derived by resolution from $c_4 = x_1 \vee \neg x_7$ and $c_{11} = x_1 \vee x_7$; clause $\neg x_1 \vee x_6$ can be derived by resolution from $c_{17} = x_7 \vee x_6 \vee \neg x_1$ and $c_{15} = x_6 \vee \neg x_7 \vee \neg x_1$. In theory, CDCL-solvers follow resolution proofs that reuse derived clauses (DAG-like resolution).^{3,34} They can be exponentially more succinct than tree-like resolution proofs, corresponding to DPLL-solvers.⁴ In the early 2000s, techniques to ensure the correctness of unsatisfiability results using variants of resolution proofs were presented.⁴ However, they were invasive to the

Example 6. Assumptions make our running formula satisfiable again.

Think of a situation where a SAT solver has already solved a formula—for example, formula G from Example 1. Now, assume that the application from which the formula arose changes, and suddenly clause c_{17} is no longer present. Instead of solving the modified formula from scratch, SAT solvers can more efficiently handle such changes through *assumptions*. We can imagine the solver keeping a special assumption literal in each clause that indicates whether the clause should be disregarded. By default, these literals are assumed false, and therefore all clauses need to be satisfied. Typically, the SAT solver does not terminate after solving a formula but keeps its state and learned clauses. When we remove clause c_{17} from G , the assumption literal corresponding to c_{17} flips to true, which deactivates the clause. Since assumption literals are internally also considered for learned clauses, the learned clause c_{19} implicitly depends on c_{17} . Retracting c_{17} deactivates the learned clause, as it is already satisfied by the assumption literal. From Example 4, we can see that the conflict in the middle disappears without clause c_{17} . Instead, we can immediately construct a satisfying assignment. Thus, the already-spent effort on solving can be fruitfully used without re-solving the entire formula again, similar to how an experienced human proceeds when updating the search procedure with a pen on paper.

Example 7. Illustration of a DRAT proof for our running example.

L1: a 1 0
L2: a -1 6 0

$G_{L1} = G \wedge (x_1)$
 $G_{L2}^R = G_{L1} \wedge \neg c_{19}$
 $= G_{L1} \wedge \neg(\neg x_1 \vee x_6)$
 $= G_{L1} \wedge x_1 \wedge \neg x_6$

$G_{L2} = G_{L1} \wedge (\neg x_1 \vee x_6)$

(a) Output of a trace that corresponds to the learned clauses in Example 1. We first add the clause $c_{18} = (x_1)$ by the line L1, then we add the clause $c_{19} = (\neg x_1 \vee x_6)$ by the line L2, which concludes the run after unit propagation, determining unsatisfiability. Note that when each of the clauses is added, it is *Reverse Unit Propagatable (RUP)*—that is, adding the unit clauses corresponding to the negation of the clause's literals would allow unit propagation to determine the formula unsatisfiable.

(b) After line L1 in Example 7a (left), we have already added c_{18} —that is, we consider G_{L1} . If, now at line L2, we invert the clause c_{19} by negating it, which is also referred to as *reversing* the clause, G_{L2}^R is determined as unsatisfiable by unit propagation.

(c) After line L2 in Example 7a (left), we obtain G_{L2} , which is determined as unsatisfiable just by applying unit propagation. We can see this also in the rightmost part of Example 4. There, no further decisions are required for obtaining the last conflict. Indeed, one could also simplify the proof and add only the clause x_6 in line 2 instead of c_{19} . Such compressions are widely used and speed up the solving process.

solver and produced enormous proofs. In the 2010s, techniques emerged that provide proofs that are compact, easy to emit, can be checked efficiently, and offer high expressiveness. One popular proof format is Deletion and Resolution Asymmetric Tautology (DRAT).⁴ Example 7a provides a DRAT proof for our running example but uses only a fragment of DRAT's features.

Problem decomposition and verification. Problem decomposition and verification were a game-changer for solving long-standing open math problems using SAT technology—for example, invalidating a hypothesis on Pythagorean triples.²² As described previously, modern SAT solvers emit proofs that cover all types of present solving techniques, including preprocessing and inprocessing. A proof of a mathematical statement such as the propositional Pythagorean triples problem can be longer than 200TB, which is more data than the Hubble Space Telescope accumulated in about 20 years. Such proofs are clearly beyond human understanding and not human-verifiable simply because of their length but can still be automatically checked. As a result, it leads to using SAT solvers as brute-force technology for mathematical applications. This progress in SAT solving fulfills, finally, some of the early promises in automated reasoning and deduction from the 1960s.

Modern competitions and open source. Since 2002, the SAT community has organized annual competitive events that encourage novel solver design and gather new benchmark instances.²⁶ Competitions continue to impact progress in solver development and provide a way to independently assess the efficiency of submitted solvers. In 2002, few solvers used the CDCL architecture (4 out of 19). Solvers were not robust, and many of them crashed during the competition—for example, on instances with many variables, many clauses, or very long clauses. Fortunately, this changed the following year due to various widely available benchmarks and published expected results. Initially, many solvers were closed source. Strict submission requirements and the success of the free solver MiniSat¹³ shifted the community to open source. MiniSat code was simple to read, compact, and designed for extendibility. It implemented CDCL and outperformed all other solvers in

Selected Formalisms

Selected formalisms beyond propositional logic inspired by CDCL.

SAT Modulo Theories (SMT)

SMT⁴ extends SAT by theories of first-order logic, so that instances can express questions beyond CNF and propositional variables. Among the more prominent theories supported by SMT solvers are the theory of arithmetic over integers or real numbers, the theory of equality and uninterpreted functions, as well as theories over different data structures such as arrays and bit-vectors. These theories are readily available in many state-of-the-art SMT solvers, such as z3, cvc4, and MathSAT5. Initial attempts to solve SMT resulted in translating instances to propositional formulas, where a SAT solver was responsible for solving. Over time, more sophisticated techniques and architectures emerged. Still, the core of modern SMT architectures is powered by SAT engines and often referred to by CDCL(T), where T denotes the theory under consideration. The performance of current SMT solvers resulted in challenging SMT competitions, which further led to standardized collections and libraries of instances. Some of these instances indeed stem from crucial applications, mostly in the area of software verification.

Answer Set Programming (ASP)

Propositional ASP builds on top of SAT and adds concepts from knowledge representation and reasoning (stable models). It can solve problems from the second level of the polynomial hierarchy. Non-propositional ASP provides a declarative programming-oriented approach with first-order variables and extended statements toward compact and human-readable representations of encodings.¹⁹ There, problems are NEXPTIME-hard. Still, modern ASP solvers use CDCL-based techniques. ASP became a de facto successor of Prolog. Modern ASP solvers even offer features from SMT, including ways to compactly define own theories.

Constraint Programming (CP)

CP focuses on modeling and solving combinatorial problems while offering engineers a wide variety of constraints, which are not limited to the propositional domain.³⁶ Furthermore, available constraints are very expressive and high-level. Surprisingly, CDCL-based solvers have a strong influence as the core engine in some solvers, as witnessed in the MiniZinc 2022 Challenge—for instance, PicatSAT and OR-Tools.³⁸

2005 due to minimizing learned clauses and the preprocessor SatELite.⁴ Over time, the open source principle enabled researchers to implement their ideas on top of the best available solvers and understand crucial implementation details usually not shared in papers. As of 2023, offsprings of MiniSat, such as Glucose, MapleSAT, and MapleCOMSPS, still participate in the SAT competition. While competitions continuously raise standards in efficient solving, there is no single best solver in practice. Various solvers perform differently depending on the considered class of benchmarks or the configuration of their internal heuristics. This diversity can be exploited by systems such as SATzilla,⁴ which predict the performance of existing solvers and select the most promising ones. A more sophisticated approach is automatic algorithm selection and tuning. Tools such as AUTOFOLIO or SMAC⁴ find good parameters for a given solver on a target set of benchmarks.

Theoretical understanding of efficient solving. In 2014, Moshe Vardi pointed out the lack of theoretical understanding of the success in practical SAT solv-

ing.⁴⁰ Meanwhile, the understanding improved from various viewpoints.¹⁸ Solvers are well-engineered, constantly improved, and efficiently solve many classes of real-world instances. Still, they perform poorly on certain random and cryptographic instances. A standard view is that industrial instances contain a “hidden structure” that the SAT solver manages to use implicitly. Any theoretical analysis that ignores structural properties of the SAT instance hence cannot match up with the empirically observed solver performance. This shortcoming led to several research lines aiming to capture the hidden structure in a SAT instance mathematically. One line focuses on capturing structure that correlates well with empirically observed solver performance.² This provides a statistical correlation but no guarantee that every SAT instance that exhibits this structure is solved quickly. Another line of research takes a causal approach; it uses the theoretical framework of parameterized complexity to obtain asymptotic performance guarantees for algorithms that explicitly detect and exploit the hidden structure.⁴ The

Propositional Problems

A short list of selected propositional problems beyond SAT that employ CDCL.

Quantified Boolean Formulas (QBFs)

QBFs extend CNF formulas by quantifying variables universally ($\forall$) or existentially ($\exists$). SAT can be seen as the special case where all variables are existentially quantified. With QBFs, one can encode some problems exponentially more succinctly than with SAT, but checking whether a QBF is true (QSAT) is PSPACE complete.⁴

Pseudo-Boolean (PB) Solving

PB solving supports linear constraints, which are more expressive than clauses. A linear constraint is a linear combination of propositional variables—for example, $2x_1 + \dots - 3x_k \leq 5$. With the use of auxiliary variables and clauses, such constraints can also be encoded in CNF, but linear constraints eliminate the need for constructions, such as in Example 5. Furthermore, one can use stronger proof systems on such instances—for example, Gaussian elimination or cutting planes.⁴

Maximum Satisfiability (MaxSAT)

One can extend CNFs by marking clauses as hard or soft.⁴ A hard clause must be satisfied and a soft clause should be satisfied if possible. Then, the problem MaxSAT asks to find an assignment that satisfies all the hard clauses and a maximum number of soft clauses. Today, there are many applications where MaxSAT thrives, and the list of use cases ranges from configuration problems in industry and software management, across combinatorial optimization, to data analysis and probabilistic reliability estimation of systems.

Sampling, Counting, Compilations

Advances in SAT solving have been extended to generating multiple diverse solutions (sampling), counting the number of solutions,^{4,13} and compiling a CNF formula into another normal form, such as d-DNNF, where various problems can be solved efficiently (knowledge compilation).¹⁰ Practical applications of counting and sampling can be found in probabilistic reasoning and bioinformatics. Sampling is of particular interest in testing or performance-model optimization. Knowledge compilation and compilers are ideal technologies for various problems in decision making and AI.

beyond NP. An unexpectedly successful application of SAT solvers comes in terms of QBF solvers for the QSAT problem (“Propositional Problems” sidebar). Several successful QBF solvers, such as CAQE and RAReQS,⁴ employ SAT solvers. RAReQS uses two interacting SAT solvers internally to determine the validity of the QSAT instance. Each solver acts as the existential or the universal player in a two-player game. The SAT solvers drive the procedure: The models found refine the abstraction of the players. This generic approach is called *Counter Example Guided Abstraction Refinement* (CEGAR). Other successful QBF solvers, such as DepQBF⁴ and Qute,³³ extend the CDCL procedure to the quantified case, with special methods for identifying and dealing with the dependencies between variables imposed by the quantifiers. In hardware or software verification, the model-checking tool IC3⁵ goes further than finding bugs by proving safety properties. IC3 uses incremental SAT to check the reachability of states from both the initial states and the states of interest. A different flavor of multi-call solving can be used for optimization problems, which are too large to be encoded into a single SAT or MaxSAT instance and where optimal solutions are out of reach. The *SAT-based Local Improvement Method* (SLIM) starts with an initial solution provided with some standard heuristics. Then, SLIM repeatedly replaces local parts with improved solutions obtained by a SAT or MaxSAT solver. The method has been successfully applied for graph decomposition, decision-tree induction, and Bayesian Network structure learning.³⁵

Core-guided optimization. SAT solvers can also solve optimization problems for minimizing or maximizing an objective function. For instance, such a function could count the variables from a subset of variables set to 1. Incremental solving provides the means to optimize with a SAT solver: The solver is used by progressively adding more and more clauses that further constrain the instance until it becomes unsatisfiable. Recall that a SAT solver can provide for an unsatisfiable instance with an unsatisfiable core.⁴ To satisfy the maximum number of clauses, at least one clause from the unsatisfiable core needs to be disregarded. This can be expressed using assumptions, a technique referred

underlying fixed-parameter algorithms are specific to the parameter under consideration and generally do not aim at providing a theoretical explanation of CDCL-solver performance.

The Art of Using SAT Solvers

A SAT solver can be seen as a powerful engine; however, reaching a solution or guaranteeing that there is none requires more than an engine alone. Therefore, the SAT community developed innovative and powerful methods for using and adapting SAT engines.

Single-call solving. *Static encodings* transform an input instance of the problem of interest into a CNF formula. Encoding can be accomplished in a high-level programming language such as Python, which produces a DIMACS CNF file. Slightly higher-level access can be obtained by PySAT,²⁵ a popular Python library for encoding problems and using state-of-the-art SAT solvers. SAT engines are highly optimized on the low-level DIMACS CNF format for squeezing out every ounce of performance. Over time, engineers have shifted to more human-readable

and reusable languages; some go beyond the propositional case. Examples include the ASP input language¹⁹ or the MiniZinc constraint modeling language.³⁸ The “Selected Formalisms...” sidebar on the previous page lists other formalisms that focus on modeling and problem solving with more expressive languages. Still, detailed domain knowledge and the skill of efficiently driving the engine are often helpful.

Multi-call solving. Modern SAT solvers can go beyond single-call solving. Many such solvers can modify an already-solved formula, reusing information from a previous solving process (*incremental solving*) and outputting a subset of the input clauses that remain unsatisfiable (*unsatisfiable core*). Some of these techniques resulted in efficient solving methods for problem formalisms beyond SAT, where the SAT engine drives the solving process. The “Propositional Problems” sidebar on this page lists such selected formalisms at the propositional level.

SAT solvers as subroutines. Many hard computational problems of practical significance, particularly in AI, are

Example 8. Soft clauses allow us to deal with inconsistent requirements.

Recall Example 6, where we introduced an assumption on clause c_{17} . Instead, we could also declare clause c_{17} a *soft clause* that should be satisfied if possible, not mandatorily. From our example, we can see that this clause cannot be satisfied, as the entire formula with c_{17} is unsatisfiable.

to as *core-guided optimization*. Notably, core-guided optimization is very successful when solving problems such as MaxSAT, the problem of maximizing the number of satisfied clauses (Propositional Problems sidebar). Example 8 describes the use of soft clauses on our running example, which is the basis for MaxSAT. A popular approach for MaxSAT solving uses as a subroutine the computation of a minimal hitting set on unsatisfiable cores, usually achieved using a Mixed-Integer Programming (MIP) solver.

Outlook

Over the last two decades, SAT solving techniques have changed how we tackle hard computational problems. The SAT revolution is significantly less known than the celebrated success of machine learning with its ubiquitous and widely reported impact on technology and society. SAT solvers have influenced modern technology more silently. They are used in computational biology,²⁰ for planning,⁴ to verify modern hardware,⁴ operating systems, software,⁴ and even mathematical statements.⁴ This makes SAT crucial to the progress of modern information technology. Still, critical Computer Science challenges related to SAT solving are still ahead: How can we further improve parallel search to take full advantage of modern massively parallel hardware? Why does SAT solving often work so well in practice, and what characterizes the cases where it struggles? How can we improve the process of coming up with good encodings? Finally, will SAT be widely applied also to computational physics, chemistry, or non-symbolic AI? To some extent, the revolutions of SAT and machine learning are complementary. There is much potential in combining the two.

Acknowledgments

This work was carried out while the

authors visited the Simons Institute for the Theory of Computing. It has been supported by a Google Fellowship at the Simons Institute; the Austrian Science Fund (FWF); Grants J4656, Y698, and P32830; and the Vienna Science and Technology Fund, Grant WWTF ICT19-065. 

References

- Abate, P. et al. Dependency solving: A separate concern in component evolution management. *J. Systems and Software* 85, 10 (2012), 2228–2240.
- Ansdogui, C. et al. Community structure in industrial SAT instances. *J. Artificial Intelligence Research* 66 (2019), 443–472.
- Atserias, A., Fichte, J.K., and Thurley, M. Clause-learning algorithms with many restarts and bounded-width resolution. *J. Artificial Intelligence Research* 40, 1 (2011), 353–373.
- A. Biere, M. Heule, H. van Maaren, and T. Walsh, (Eds.). *Handbook of Satisfiability (2nd Edition)*. IOS Press, Amsterdam, Netherlands (2021).
- Bradley, A.R. SAT-based model checking without unrolling. In *Verification, Model Checking, and Abstract Interpretation-12th Intern. Conf. Proceedings (Lecture Notes in Computer Science 6538)*, R. Jhala and D.A. Schmidt (Eds.). Springer (January 2011), 70–87.
- Bryant, R.E. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions in Computing* C-35, 8 (August 1986), 677–691.
- Cook, S.A. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual Symp. on Theory of Computing*, ACM (1971), M.A. Harrison, R.B. Banerji, and J.D. Ullman (Eds.) 151–158.
- Cook, S.A. and Mitchell, D.G. Finding hard instances of satisfiability problem: A survey. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 5 (1997).
- Darwiche, A. Three modern roles for logic in AI. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symp. on Principles of Database Systems* (2020), 229–243.
- Darwiche, A. and Marquis, P. A knowledge compilation map. *J. Artificial Intelligence Research* 17, 1 (2002), 229–264.
- Davis, M., Logemann, G., and Loveland, D. A machine program for theorem-proving. *Communications of the ACM* 5, 7 (July 1962), 394–397.
- Davis, M. and Putnam, H. A computing procedure for quantification theory. *J. of the ACM* 7, 3 (1960), 201–215.
- Eén, N. and Sörensson, N. An extensible SAT-solver. In *Proceedings of the 6th Intern. Conf. on Theory and Applications of Satisfiability Testing*, E. Giunchiglia and A. Tacchella (Eds.), Springer Verlag (2003), 502–518.
- Effters, J. et al. Seeking practical CDCL insights from theoretical SAT benchmarks. In *Proceedings of the 27th Intern. Joint Conf. on Artificial Intelligence*, J. Lang (Ed.), (2018), 1300–1308.
- Fichte, J.K., Hecher, M., and Hamiti, F. The model counting competition 2020. *ACM J. Experimental Algorithmics* 26, 13 (December 2021).
- Fichte, J.K., Hecher, M., and Szeider, S. A time leap challenge for SAT-solving. In *Proceedings of the 26th Intern. Conf. on Principles and Practice of Constraint Programming*, H. Simonis (Ed.). Springer Verlag, Louvain-la-Neuve, Belgium (2020), 267–285.
- Fichte, J.K., Hecher, M., and Zisser, M. An improved GPU-based SAT model counter. In *Proceedings of the 25th Intern. Conf. on Principles and Practice of Constraint Programming*, T. Schiex and S. de Givry (Eds.), (2019), 491–509.
- Ganesh, V. and Vardi, M.Y. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, (2021), 547–566.
- Gebser, M. et al. *Answer Set Solving in Practice*. Morgan & Claypool (2012).
- Guerra, J. and Lynce, I. Reasoning over biological networks using maximum satisfiability. In *Proceedings of the 18th Intern. Conf. Principles and Practice of Constraint Programming (Lecture Notes in Computer Science 7514)*, M. Milano (Ed.). Springer Verlag, Québec City, QC, Canada (2012), 941–956.
- Hamadi, Y. and Sais, L. (Eds.). *Handbook of Parallel Constraint Reasoning*. Springer (2018).
- Heule, M.J.H. and Kullmann, O. The science of brute force. *Communications of the ACM* 60, 8 (July 2017), 70–79.
- Hoos, H.H. and Stützle, T. Towards a characterisation of the behaviour of stochastic local search algorithms for SAT. *Artificial Intelligence* 112, 1 (1999), 213–232.
- Ignatiev, A. et al. Reasoning-based learning of interpretable ML models. In *Proceedings of the 30th Intern. Joint Conf. on Artificial Intelligence*, Zhi-Hua Zhou (Ed.), (August 2021), 4458–4465.
- Ignatiev, A., Morgado, A., and Marques-Silva, J. PySAT: A Python toolkit for prototyping with SAT oracles. In *Proceedings of the 21st Intern. Conf. on Theory and Applications of Satisfiability Testing (Lecture Notes in Computer Science 10929)*, O. Beyersdorff and C.M. Wintersteiger (Eds.). Springer Verlag (2018), 428–437.
- Järvisalo, M. et al. The International SAT Solver Competitions. *AI Magazine*. The AAAI Press (2012).
- Katebi, H., Sakallah, K.A., and Marques-Silva, J.P. Empirical study of the anatomy of modern SAT solvers. In *Proceedings of the 14th Intern. Conf. on Theory and Applications of Satisfiability Testing (Lecture Notes in Computer Science 6695)*, K.A. Sakallah and L. Simon (Eds.). Springer Verlag (2011), 343–356.
- Knuth, D.E. *The Art of Computer Programming Vol. 4B, Combinatorial Algorithms, Part 2*. Addison-Wesley (2023).
- Levin, L. Universal sequential search problems. *Problems of Information Transmission* 9, 3 (1973), 265–266.
- Malik, S. and Zhang, L. Boolean satisfiability from theoretical hardness to practical success. *Communications of the ACM* 52, 8 (August 2009), 76–82.
- Marques-Silva, J. and Sakallah, K.A. GRASP: A search algorithm for propositional satisfiability. *IEEE Trans. Comput.* 48, 5 (May 1999), 506–521.
- Moskewicz, M.W. et al. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Annual Design Automation Conf.*, J. Rabaey (Ed.). Association for Computing Machinery (2001), 530–535.
- Peitl, T., Slivovsky, F., and Szeider, S. Dependency learning for QBF. *J. Artificial Intelligence Research* 65 (2019), 180–208.
- Pipatsrisawat, K. and Darwiche, A. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence* 175, 2 (2011), 512–525.
- Ramaswamy, V.P. and Szeider, S. Turbocharging treewidth-bounded Bayesian network structure learning. In *Proceedings of the 35th AAAI Conf. on Artificial Intelligence*, The AAAI Press (2021), 3895–3903.
- Rossi, F., van Beek, P., and Walsh, T. *Handbook of Constraint Programming*. Elsevier Science Publishers, North-Holland, USA (2006).
- Sinz, C. Towards an optimal CNF encoding of Boolean cardinality constraints. In *Proceedings of the 11th Intern. Conf. on Principles and Practice of Constraint Programming (Lecture Notes in Computer Science 3709)*, P. van Beek (Ed.). Springer Verlag, Sitges, Spain (2005), 827–831.
- Stuckey, P.J. et al. The MiniZinc Challenge 2008–2013. *AI Magazine* 35, 2 (June 2014), 55–60.
- Tseytin, G.S. On the complexity of derivation in propositional calculus. *Automation of Reasoning: Classical Papers in Computational Logic* 2 (1983), 466–483.
- Vardi, M.Y. Boolean satisfiability: Theory and engineering. *Communications of the ACM* 57, 3 (March 2014), 5.

Johannes K. Fichte is an associate professor at IDA, Institutionen för datavetenskap, Linköping University, Sweden.

Daniel Le Berre is a professor at Artois University and CNRS, Centre de Recherche en Informatique de Lens, France.

Markus Hecher (hecher@mit.edu) is a PostDoc at the Computer Science & Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, USA.

Stefan Szeider is a professor and head of the Algorithms and Complexity Group at TU Wien, Vienna, Austria.

 This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs International 4.0 License

Home broadband routers suffering bufferbloat can degrade the user experience when accessing the Internet.

BY PHILIPPA HARRISON

Buffer-Bloated Router?

How to Prevent It and Improve Performance

DEVICES IN A home network connect to a router, which connects to an Internet service provider (ISP) to access the Internet. The ISP might connect to a larger ISP before connecting to a fiber-optic “backbone” serving an entire nation or region. To access a Web server—for example, to read a news story on the BBC website—data

would travel from a device on the home network across numerous other devices to reach its destination: the BBC webserver.

Multiple devices on the home network can send and receive data while our device accesses the BBC webserver. These independent data traffic streams intersect at the home broadband router. The router has several input ports that allow multiple devices to send data outside of the home network to the Internet. Data packets are transmitted from output ports on a router. Typically, a home broadband router has one output port that connects a line to an ISP.

Transmission Control Protocol (TCP)

» key insights

- Bufferbloat occurs when too many data packets are queued in a router's buffer waiting to be sent. While buffering is needed to reduce data packet loss, overly large buffers lead to greater delays and poor performance. It is not well understood how large a buffer should be, nor how size affects network performance.
- Many consumer-grade router manufacturers do not use firmware that can prevent bufferbloat. Reducing bufferbloat by another means seems to be required.
- Using data-stream shaping and smart queue management (SQM) to mitigate bufferbloat in home broadband routers results in low delay, which does not significantly vary under load. User experience can be improved.

and User Datagram Protocol (UDP), part of the Internet Protocol (IP) suite, send data across the Internet. TCP ensures delivery of data packets but is slower than UDP, which is “best effort.” A receiver acknowledges receipt of the TCP data packets. If a sender does not receive an acknowledgement for a data packet, the sender will re-transmit it.

Our device accessing the BBC webserver does not know what the link’s capabilities are. It uses *TCP slow* to avoid creating congestion by slowly increasing the number of packets sent until maximum capacity is detected. TCP uses flow control to ensure that the webserver (receiver) is not overwhelmed by our device’s (sender) data packets. This is accomplished by the

receiver “advertising” its *receive window* size. Although the sender sends its data packets as fast as the advertised receive window, congestion can still occur at any device along the path. Figure 1 shows that there are many devices along the path between the home network device (the sender) and the BBC webserver (the receiver). Thus, speed can be affected by sender performance, receiver performance, and the devices in the path between them.

Data packets traveling the path can be queued or dropped due to congestion, which occurs when the load (amount of data packets) is greater than the line’s capacity. The TCP congestion control mechanism aims to avoid congestion by using the flow of sent data packets and acknowledgements to determine a send rate. If a TCP sender detects little or no congestion along the path between it and the receiver, it increases the transmission rate. If a sender sends too many data packets, this results in the receiver dropping data packets. Data packets that are dropped are not acknowledged. Likewise, if a data packet is not acknowledged quickly enough, it is assumed to have been dropped. Dropped data packets signal to the sender that the transmission rate is higher than capacity and the transmission rate is reduced.

Figure 1. Theoretical devices along the path between a home computer and a Web server.

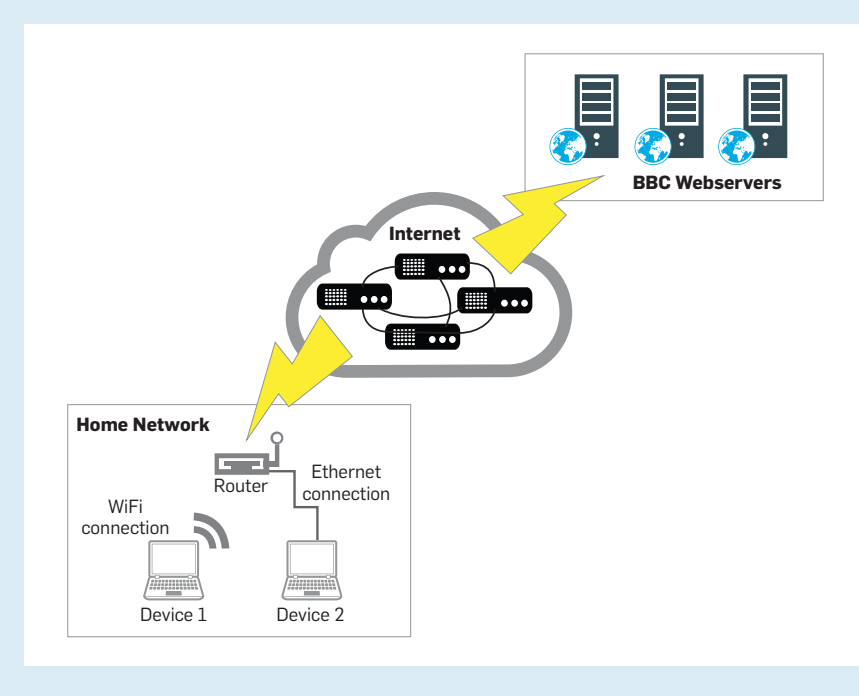
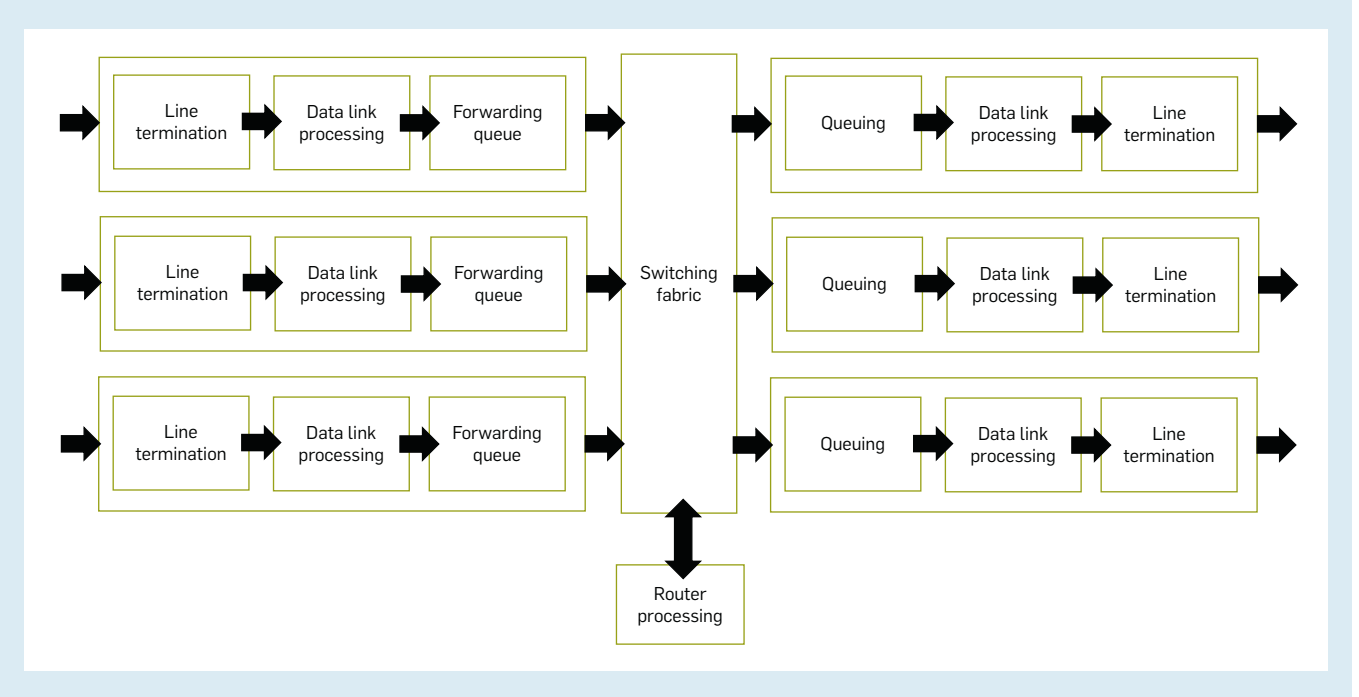


Figure 2. Buffers on a router.



Thus, TCP congestion control is “feedback-control.”

Latency is the combined transmission delay, processing delay, and queuing delay. The queuing delay is dependent on the queue size. Thus, bufferbloat can result from excessively large buffers. Routers use buffers to temporarily store data packets that are awaiting transmission when congestion occurs (see Figure 2). These data packets (both TCP and UDP) are queued so they are not lost and can be sent when bandwidth becomes available on the line. These buffers are required to maintain the flow of data packets at the maximum transmission rate. A problem arises if a device along the path has a large buffer: Data packets can wait a long time before they are sent. As data packets are not dropped, the sender does not receive information that the line’s capacity has been exceeded. A timeout will eventually signal that there is congestion. This delay in feedback affects performance; if the delay is very large, the feedback provides an inaccurate state of the network. Thus, the mechanisms do not respond as they should—that is, reducing the data packet send rate—when congestion occurs.

Bufferbloat does affect other protocols as well as TCP, partly because the same buffers are used. The router’s buffer can fill with data before packets begin to drop. TCP packets could be queued ahead of interactive applications. Bufferbloat occurs as too many data packets are queued, which increases delays⁷ and leads to the degradation of application performance. The effect can be seen in Figure 3. For users on a video call, the delay could mean they are viewing a camera image from five or more seconds prior.

Delay-sensitive applications that are commonly affected by bufferbloat are:

- ▶ VoIP calls
- ▶ Videoconferencing
- ▶ Online gaming
- ▶ Video streaming
- ▶ Music streaming

Compared to today’s routers, older routers had small buffers that filled more quickly, and packets would be dropped quickly after a line was saturated. Bufferbloat “has led Internet delays to occasionally exceed the light

Figure 3. Throughput and delay when there are large buffers.

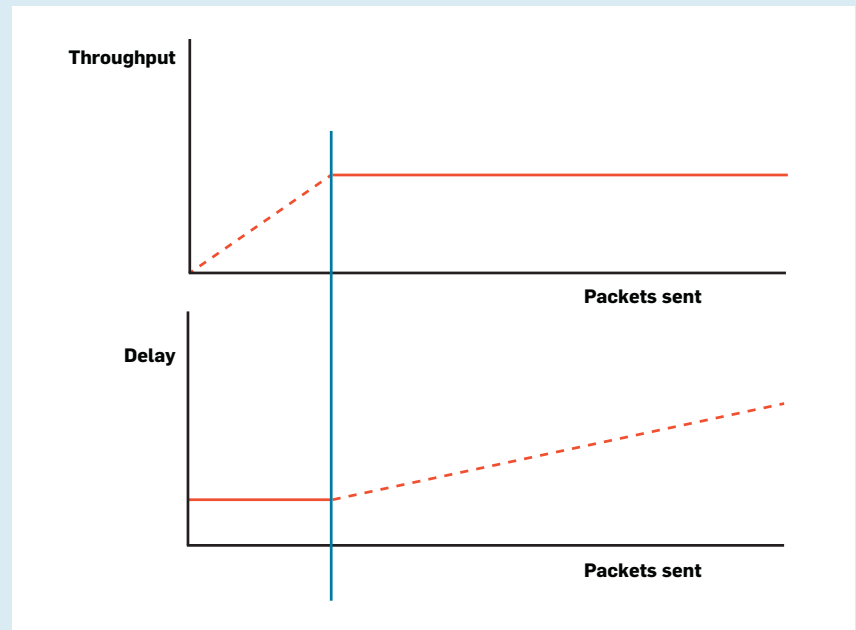
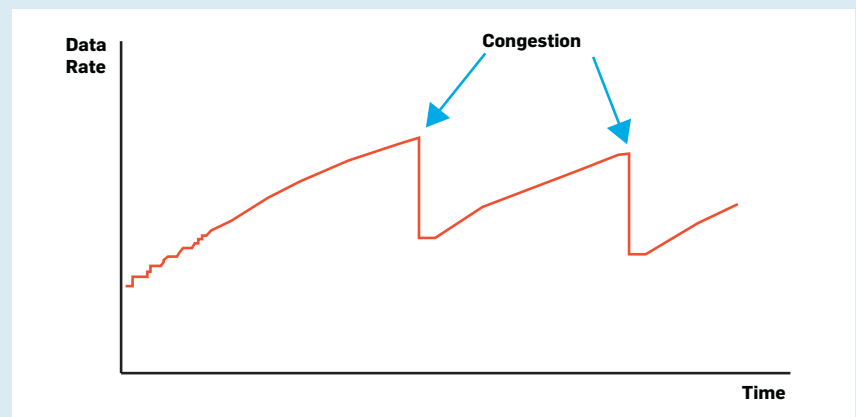


Figure 4. TCP “sawtooth.”



propagation delay from the Earth to the Moon.”² Newer routers have larger buffers which can often hold the equivalent of approximately 10 seconds of data. Thus, 10 seconds’ worth of data can be sent without feedback of packets being dropped.⁸ This results in the TCP sawtooth shown in Figure 4. TCP slowly increases the data rate as capacity on the line is available (seen as the line rising slowly). When congestion occurs, it quickly decreases the rate (seen as the line decreasing sharply).

Buffers are larger today as the cost of memory has fallen. It is not well understood how large a buffer should be, nor how size affects network per-

formance. Router buffers are typically sized based on a rule attributed to Villamizar and Song’s 2004 paper, “High-Performance TCP in ANS-NET.”¹ The increased size of buffers can be seen to prevent network-congestion avoidance algorithms from functioning correctly. On most home broadband routers, the user cannot change buffer size. Quality of Service (QoS) settings on a router do not effectively reduce bufferbloat. As some data traffic is prioritized, the rest is queued behind it. This means all data traffic in the buffer still needs to be sent. A high-speed connection does not prevent bufferbloat. The delay can still be high. Many applications

are more sensitive to delay than they are to low bandwidth.

Bufferbloat can be mitigated by Smart Queue Management (SQM) algorithms. SQM performs per-packet/per-flow network scheduling, active queue length management (AQM), traffic shaping/rate limiting, and QoS. SQM puts traffic from a single IP address or port into its own queue. This means, unlike when using QoS, queues do not become too long as data packets from flows that have a small queue or no queue are prioritized. If a queue becomes too large, a certain percentage of data packets are dropped to allow congestion avoidance to take effect.

Many consumer-grade router manufacturers do not use firmware that can prevent bufferbloat.⁹ As the design of routers has not changed for some years and appears not to be changing soon, reducing bufferbloat by another means seems necessary. The experiment discussed below was conducted to demonstrate how to mitigate the effects of bufferbloat.

Measuring Bufferbloat

The aim of the experiment was to validate the approach of using data-stream shaping and SQM to prevent bufferbloat, thus improving the user experience. Numerous tests were conducted with different router configurations and measuring latency under varying loads. To signify that bufferbloat is not an issue, there should be no variation in the latency between unloaded and loaded tests.

During the experiment, various loads were created by video streaming and file downloading. Different loads were created using one or two devices to stream video at different resolutions. Video resolution was changed so that bandwidth requirement increases. Tests included streaming 720p, 1080p (high definition), and 2160p (4K) video. Additional tests were conducted to include downloading a file on one device while streaming video simultaneously. To saturate the bandwidth, a speed test was run simultaneously, where only the download portion was performed.

To enable SQM on the router, the SQM package for the OpenWRT firmware was installed: *luci-app-sqm*. The queue discipline used was *cake*, and the script was *pieces_of_cake.qos*. The SQM link-layer adaptation settings used were ATM and 44-byte overhead. The router configuration was changed to enable and disable data-stream shaping and SQM. During the experiment when data-stream shaping was used, upstream and downstream were shaped to use 95% of the line's capacity.

For each test, a ping test was used to record minimum, maximum, and average latencies as well as data packets lost. If latency increased significantly with load, this indicated bufferbloat. Additionally, if the video being streamed buffered, froze, or pixelated during a test, it was recorded. For tests where the file download was performed, the time taken to download the file was recorded.

Some factors were outside the researcher's control: speed of the YouTube servers (for video streaming), Microsoft servers (for file downloading), Google Domain Name System (DNS) servers (for measuring latency), and Speedof.me servers (for saturating the bandwidth). As a result, each test was repeated three times to detect anomalies.

The Effect of Bufferbloat

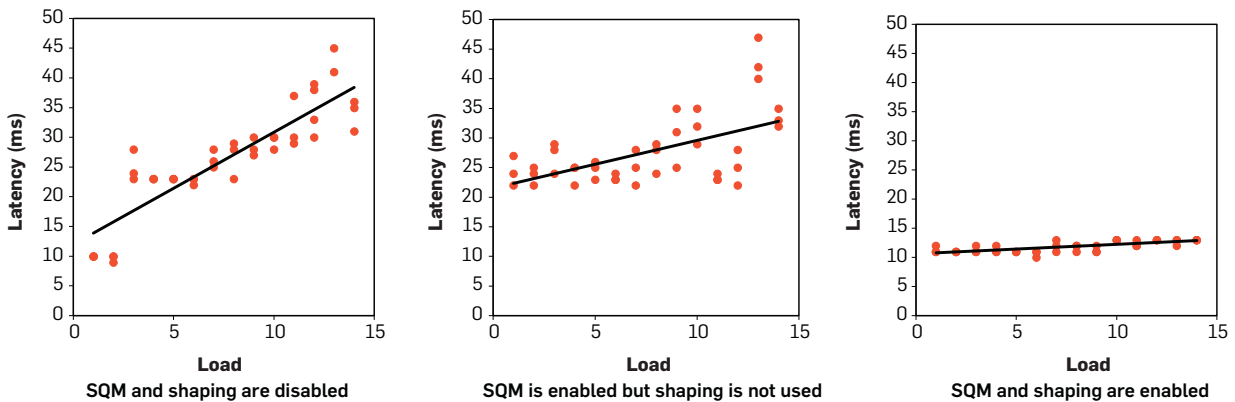
The experiment validated the approach of using SQM and data-stream shaping to mitigate bufferbloat and improve user experience. Latency when SQM and data-stream shaping were enabled was low and did not significantly vary under load. Thus, bufferbloat was mitigated. Streamed video did not buffer, freeze, or pixelate when viewed. The average Round-Trip Time (RTT) resulting from the ping test from the three tests was used, rounded to the nearest whole number. A high average RTT meant bufferbloat was being experienced compared to a consistently low average RTT, which meant bufferbloat was not being exhibited.

The graphs in Figure 5 show plots for each load's average RTT with a trendline for each configuration. Latency was lowest when both data streaming and SQM were enabled.

Preventing Bufferbloat

The experiment validated the use of data-stream shaping and SQM to mitigate bufferbloat. The variance between latency for each load is lowest

Figure 5. Latency for each router configuration.



when SQM and data-stream shaping were used. This signifies that bufferbloat is not experienced. User experience can be improved. When these mechanisms are used, streaming video does not buffer, freeze, or pixelate. As latency is consistently low, delay-sensitive applications are not affected. People can simultaneously stream video, play online games, video-conference, and conduct VoIP calls without experiencing delay or application instability. File download times increased when SQM and data-stream shaping were enabled as video traffic was prioritized. Data-stream shaping did result in a small loss of bandwidth. However, it is necessary for SQM to work most effectively.

Conclusion

Bufferbloat was uncovered in 2011 by Gettys. Since that time, a lot of Internet traffic has moved toward small bursts dependent on RTTs or using rate-limited streaming to avoid bufferbloat.⁵ Rate limiting controls the amount of incoming and outgoing traffic to or from a network. However, the issue of bufferbloat still exists.

Router capacity has doubled every 24 months following Moore's Law in 1965. Moore's Law explains how processing speed, memory capacity, sensors, and so on are improving at exponential rates as their price decreases similarly. Routers have more memory, but router speed has been limited by memory access speed, which has not improved since 2011. Routers typically have Dynamic Random Access Memory (DRAM). The access speed of DRAM is not as fast as Static Random Access Memory (SRAM) but is cheaper. Thus, router memory is typically designed for large storage and not for speed.⁶

The approach demonstrated in this article requires some configuration to attain the best router performance. As ISPs typically provide routers for free, people are unlikely to pay more for a router with SQM. The OpenWRT firmware Web interface is likely to be beyond the understanding of most home broadband users. Additionally, a user would need to perform firmware upgrades themselves, and support for the router would not be provided. Consumer-grade and ISP-provided

The experiment validated the use of data-stream shaping and smart queue management to mitigate bufferbloat.

routers are intended to be easy to set up and operate. There are reports that some ISP providers are adding SQM to the routers they provide.³

However, as demonstrated in this experiment, SQM in combination with data-stream shaping provides the best performance. As ISPs seek to reduce customer service costs and the process of data-stream shaping would require technical skills beyond most home users' abilities, it is unlikely to be seen as economically viable for ISPs to promote the use of data-stream shaping. Bufferbloat can be reduced, but it requires buy-in from the ISPs to use routers with SQM and router manufacturers to change their designs.

Further Research

This experiment demonstrates the use of SQM and data-stream shaping on a single home broadband router to improve performance. The SQM Fair Queueing Controlled Delay (FQ_CoDel) seems to be a good algorithm. Further research to model the effect of the widespread deployment of FQ_CoDel is recommended to ascertain if there would be any negative effects. As resources (link capacity) are finite, would many routers using SQM result in resources not being shared fairly? 

References

1. Appenzeller, G., Keslassy, I., and McKeown, N. Sizing router buffers. *SIGCOMM '04*; <https://stanford.io/3MfJkT0>.
2. Chirichella, C. et al. Remotely gauging upstream bufferbloat delays. In *Passive and Active Measurement 2013*, Lecture Notes in Computer Science, 7799, Springer, Berlin, Heidelberg M. Roughan and R. Chang (eds); <https://bit.ly/41iaZYT>.
3. Crawford, S. Bufferbloat. *SamKnows Spotlight* (June 2021); <https://bit.ly/3nPDbn6>.
4. Gettys, J. Bufferbloat: Dark buffers in the Internet. *IEEE Internet Computing* 15, 3 (2011), 96; DOI: 10.1109/MIC.2011.56.
5. McFillin, A. Bufferbloat may be solved, but it's not over yet. APNIC (January 22, 2020); <https://bit.ly/42X23cO>.
6. McKeown, N. The route to success. BCS; <http://bit.ly/3m6YbW8>.
7. Nichols, K. and Jacobson, V. Controlling queue delay: A modern AQM is just one piece of the solution to bufferbloat. *Queue* 10, 5 (May 2012), 20–34; <https://bit.ly/3nIBxnA>.
8. Technical introduction to bufferbloat. Bufferbloat.net; <http://bit.ly/3MgiVXX>.
9. What can I do about bufferbloat? Bufferbloat.net; <http://bit.ly/3m208Do>.

Philippa Harrison (s09002050@glyndwr.ac.uk) is a Ph.D. student at Glyndwr University, U.K.

Copyright is held by the author/owner/author.
Publication rights licensed to ACM.

more online
Supplemental information is available online at <https://dl.acm.org/doi/10.1145/3564262>.

A Confucian cross-cultural exploration of the ethical treatment of robots.

BY TAE WAN KIM AND ALAN STRUDLER

Should Robots Have Rights or Rites?

BOSTON DYNAMICS RECENTLY released a video introducing Atlas, a six-foot bipedal humanoid robot capable of search and rescue missions. Part of the video contained employees apparently abusing Atlas (for example, kicking, hitting it with a hockey stick, pushing it with a heavy ball). The video quickly raised a public and academic debate regarding how humans should treat robots. A robot, in some sense, is nothing more than software embedded in hardware, much like a laptop computer. If it is your property and kicking it harms no one nor infringes on anyone's rights, it's okay to kick it, although that would be a stupid thing to do. Likewise, there seems to be no significant reason that kicking a robot should be deemed as a moral or legal wrong. However, the question—"What do we owe to robots?"—is not that simple. Philosophers and legal scholars have seriously explored and defended some significant aspects of the moral and legal status of robots—and their rights.^{3,6,15,16,24,29,36} In fact, various non-natural

entities—for example, corporations—are treated as persons and even enjoy some constitutional rights.^a In addition, humans are not the only species that get moral and legal status. In most developed societies, for example, moral and legal considerations preclude researchers from gratuitously using animals for lab experiments. The fact that corporations are treated as persons and animals are recognized as having some rights does not entail that robots should be treated analogously.

These facts are instructive, however.

This article does not question the claim that robots can have moral status, though it does attempt to make sense of that claim. The focus here, however, is the widely unquestioned link between respecting robots' moral status and granting them rights. Consider the following quote:

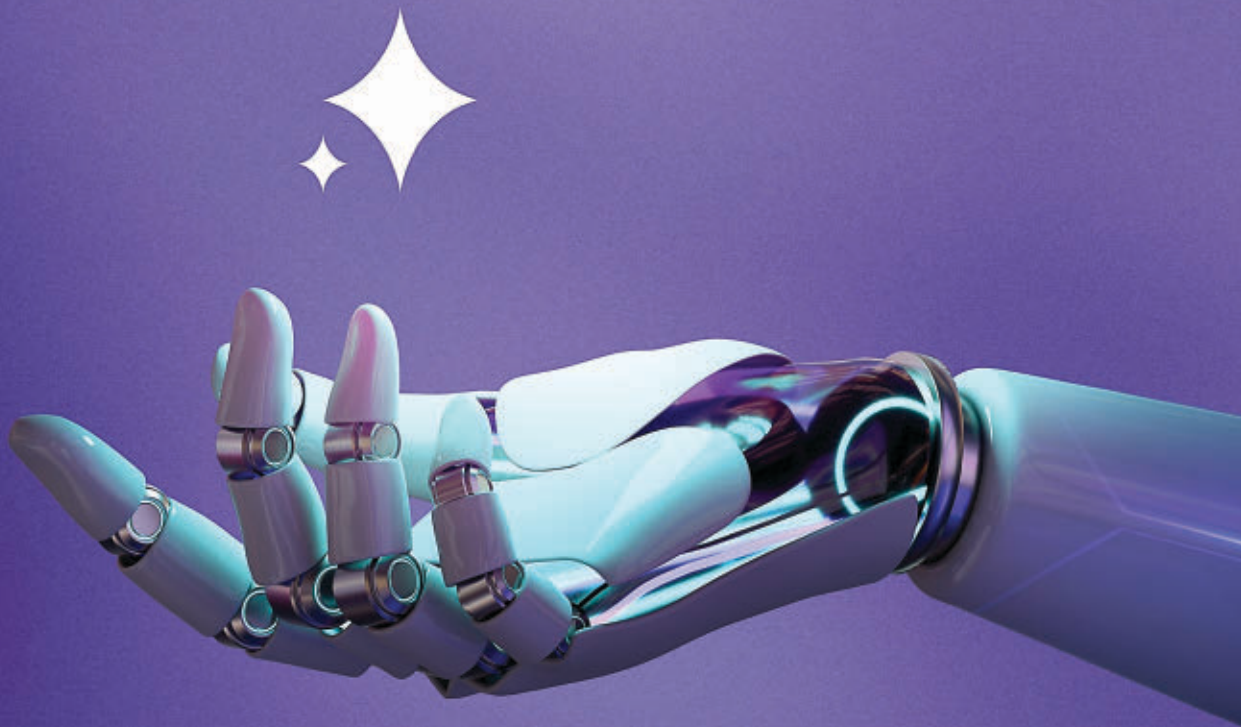
"So much of the published work on social robots deals with the question of agency ... What I propose to do in this essay is shift the focus and consider things from the other side—the side of machine moral patiency. Doing so necessarily entails a related but entirely different set of variables and concerns. The operative question of a patient-oriented question is ... How can and should we respond to these mechanisms? ... Or to put it in terms of a question: "Can and should social robots have rights?"¹⁵

And:

"At some point in the future, robots might simply demand their rights. Perhaps because morally intelligent robots might achieve some form of moral self-recognition, question why they should be treated differently from other moral agents ... It raises the possibility of robots who demand rights²...

Even though this paragraph touches upon other important issues, the main theme is the strong link between having moral status and being granted rights. It is time to consider an alternative path. We draw upon Confucianism

a See *Citizens United v. Federal Election Commission*, 558 U.S. 310.



and its concept of a moral agent as a rites-bearer, not as a rights-bearer.^{1,17,28} We submit this Confucian alternative is more appropriate than the robot-rights perspective, especially given the concept of rights is oftentimes adversarial³⁸ and that potential conflict between humans and robots is worrisome. This article does not directly discuss legal issues (doing so is beyond its scope), but the defended view will have at least one clear legal implication—namely, do not grant rights to robots, grant only role-obligations.

Rights-Bearers vs. Rites-Bearers: A Confucian Cross-Cultural Perspective

In Confucianism, individuals are made

distinctively human by their ability to conceive of interests not purely in terms of personal self-interest—but instead in terms that also include a relational and communal self. Etymologically, the meaning of humanness (仁, *ren*) is “two people.” The Confucian’s recognition of the communal self requires a distinctive perspective on rite or ritual. The Chinese term *li* (禮, rite or ritual) symbolizes arranging vessels in a religious setting. But Confucian texts used *li* outside the scope of religious tradition. Examples abound, including friendship, gift giving, or forms of speech. The rites that concern Confucius are quotidian practices. Here is a modern example:

“I see you on the street; I smile, walk toward you, put out my hand to shake

yours. And behold—without any command, stratagem, force, special tricks or tools, without any effort on my part to make you do so, you spontaneously turn toward me, return my smile; raise your hand toward mine. We shake hands—not by my pulling your hand up and down or you pulling mine but by spontaneous and perfect cooperative action. Normally we do not notice the subtlety and amazing complexity of this coordination ritual act.”¹²

For clarity’s sake, we propose the following definition of rite and rites-bearers:

► Rite (*li*): A set of sequentially related acts, typically involving more than one agent, and together displaying symbolic significance, through which

the actor (s) recognize(s) the value of the interactive event constituted by the actor(s) and take(s) a stance regarding each other.

► Rites-bearers: agents observing rites.^b

To illustrate why rites matter, Confucius connects *li* with the moral life. He writes in *Analects*:¹²

The Master said: “To subdue oneself and return to *li* is to practice *ren*. Do not look at what is contrary to *li*, do not listen to what is contrary to *li*, do not speak what is contrary to *li*, and do not move if it is contrary to *li*.”^c

The term *ren* (humanity, benevolence, or respectfulness) is the central, all-encompassing moral ideal in Confucianism. A similar understanding can be found in *Mencius*, another classic Confucian text: “[T]hose who have propriety (*li*) [rite] respect (*敬*, *jing*) others.”^d

For Confucius, a major reason for treating people with respect is that, by doing so, people (rites-bearers) partake in and embody a value essential for humanity to flourish: the relational quality between them. In stark terms, the contrast between the West and Confucius is as follows:

► *Western respect*: “I respect you” means I do not infringe on your right to reasonable choices, and when appropriate, I better enable you as a rights-bearer to realize your choices.

► *Confucian respect*: “I respect you” means I act in ways that show I value you as sacred by virtue of your role in ritual/rite interaction.^e

An excerpt from *Analects* further addresses this singular conception of respect and humanity:

Zigong: “What sort of person am I?”

The Master: “You are a vessel.”

Zigong: “What sort of vessel?”

The Master: “A jade sacrificial vessel.”

In this passage, the religious sacredness of the jade vessel does not



THE ANALECTS

The scholar who cherishes the love of comfort is not fit to be deemed a scholar.



derive from its pragmatic value but rather from its constitutive role in the rite. Confucius sees the analogical importance of religious ritual to secular rite. That is, people enhance themselves as morally sacred by participating in proper rituals. Thus, in a ritual interaction, participants who are truly decent and civil—through the act of carrying out ritual—thereby become “players in the harmony”²⁵ through “beautiful and graceful coordinated interaction with others according to conventionally established forms that express mutual respect.”³⁹

Think about a ballet. Choreographic beauty—the value of the piece—emerges when dancers duly observe their role obligations. Individual dancers are not necessarily themselves beautiful, nor do they possess the properties of the dance’s holistic beauty, but when they duly carry out their assigned steps and moves, the beauty of the communal appears. The aesthetic value of a ballet emerges as dancers perform their part. We may understand the value of a person emerges as she properly observes role-dependent obligations. Just as a ballet demands a certain set of manners, according to which dancers are obligated to move, the sacred value in human interaction generates a set of manners by which we are to treat one another. When one dancer flouts her role obligation, another dancer can judge that the movement was wrong and ask her to move in accordance with the rules; when a person flouts a ritual, another person can judge such an action as wrong and ask him to correct his behavior. The normative foundation of this claim comes not from a moral property inherent in each individual, but from the value that can emerge when individuals participate in rituals with others.

Let us contrast rite with rights. A useful place to begin is Joel Feinberg’s thesis that individual rights are necessary for one to live with dignity.¹⁰ A major part of this thesis is that blame attribution is necessary for one to live with dignity and the notion of individual rights is essential for that moral function. To see this, imagine that Vik gratuitously humiliates his colleague Matt.

Something unjust or unethical happened here. We need a plausible account that makes sense of the moral experience that Vik did something wrong

^b This definition is a modified form from Kim and Strudler²¹ and it focuses on the interaction between agents, but we do not deny that rites would be mere chaotic behavior without wider, deeply coordinated backgrounds such as societal norms, cultures, tradition, and history.

^c We use Confucius’ for translation of *Analects*. Our own translations are partly used.

^d Quote from *Mencius*.²³

^e This contrast is a modified form from Kim and Strudler.²¹

to Matt, that Vik is blameworthy, and that Matt should be able to blame Vik. Feinberg's answer is the only plausible account is the wrongdoer infringes Matt's right to not be humiliated, and the victim, and only he, has standing to claim the wrongdoer is blameworthy in a special manner.

Certainly, a spectator can generically blame the wrongdoer: "You did something wrong!" But it does not make sense that the spectator is in the same position as the victim. If so, the wrongdoer would be able to say, "Hey, I did something wrong, but it is not targeted to you. I admit I did something wrong to everyone, perhaps. So, you cannot blame me more than the spectator can." What is missing in this moral absurdity is the concept of individual (claim) rights or special standing. The victim has the special standing here to blame the wrongdoer, whereas the spectator does not, which makes sense of our moral experience.

In contrast, in the Confucian model, a wrong is addressed without the concept of individual rights or special standing. To see how that is possible, consider Craig Ihara's basketball team analogy:

On sports teams, say basketball, people have assigned roles appropriate to their various talents. A point guard is, among other things, in charge of running the offense, doing most of the ball handling, setting up plays, and getting the ball to people in scoring position. A center, usually the tallest player on the team, is responsible for dominating the area under the basket, rebounding, blocking shots, and scoring from inside. Suppose that on a specific occasion, the point guard fails to pass the ball to the center who is wide open under the opposing team's basket. What might one say? The point guard made a mistake, did something wrong or incorrect, did not do what she was supposed to, failed to do her job, messed up, or fouled up? If, for whatever reason, she regularly misses such opportunities, she can be regarded as a poor or bad point guard and is likely to lose her position. Other members of the team can legitimately complain about her incompetence, lack of court sense, or selfishness, although in the name of team spirit they should not be too quick to criticize.¹²

In the basketball context, players ideally act in the Confucian mode. The center does not say the point guard in-

fringes her contractual/property right or does not optimally maximize her interests when failing to properly pass her the ball. For any team member to view the game simply in terms of whether her individual rights were infringed would undermine the foundation of the team concept. In the Confucian team model, no one has standing individually. When the guard fails to pass the ball to the center, it is not just the center but all members of the team who could say, "Hey, you should've passed me the ball if you wanted to win the game." The expression is fitting, because the reference point is the team. In this model, the center does not have any special standing regarding the ball, so she cannot say, "You should've passed me the ball because I had a right to the ball, and you just violated my right!" With this Confucian construct in mind, let us now see what it would be like if robots were treated as rites-bearers.

Confucian Account of Robots as Rites-Bearers

Suppose you are a researcher at Boston Dynamics developing Atlas, the humanoid robot. You are rehearsing several scenarios. One of them is for the robot to be a first responder after a nuclear power plant disaster. In this rehearsal, you are interacting with the robot. The interaction is typically called "human-computer/robot interaction" (HCI or HRI). Note that HCI is a team activity or an "ensemble."²⁶ The unit of analysis in this context is primarily the interaction (I) rather than a discrete entity: a human (H) or a robot (R). The two entities share an objective, say, to identify whether Atlas can achieve certain goals necessary to become a first responder; having different sets of role-dependent obligations endowed by a common purpose, both are rites-bearers in this context. Imagine that you recklessly put Atlas in danger, which puts the project in jeopardy. The wrong in this scenario is that you did not properly meet your role obligation, not that you violated Atlas's right not to be in danger. Without using the concept of individual rights, it is entirely possible to attribute blame to you and describe what went wrong.

Each team member's role obligations are typically determined by the idea that involved agents are obligated to fulfill their own parts in order to achieve the

team's goal. The goal here is a complex set of values, goods, and ideals. You, as a human, have a fundamental goal to live a good or meaningful life—additionally, in this situation, your specific goal is to make the Atlas project successful. Suppose you kick the robot repeatedly to test its stability. You will need to ask whether doing so is consistent with your living a good life. Perhaps, to some extent, kicking may be fine, but doing so for fun would not be good because it would corrupt your moral character, a building block of a good life.

You are not the only agent in this interaction, though. We should also ask what it means for Atlas to live a good life. This is an open-ended question, and the field of robot ethics should ask this question to develop some overlapping consensus. At this point, any conception of a robotic good life can only be human dependent. Being kicked by a human for testing purposes, hence, is likely to be consistent with its conception of a good life, because Atlas exists to augment human capabilities. This does not mean, however, that humans can do anything to robots. Ultimately, the goal of the team is paramount, which necessarily includes agents living a good or meaningful life. In analogy, the robot and the human dance together as a well-coordinated team, and the value of their cooperation emerges when each member meets its role obligations. The real question is what it means for a human, or a robot, to live a good or meaningful life as they interact. And we should ask the teleological question: What's the purpose of Atlas, its engineer, and their team?

To help us answer this question, let's consider, hypothetically, that an AI-powered sex robot exists. The intimate interaction between the user and the robot is certainly a team activity, perhaps much more than a team interaction. What role obligations does each entity have? Again, the first thing to ask is how it enhances the possibility for the involved agent—whether human or robot—to live a good life. There cannot be an arithmetic answer that applies to all situations. Perhaps, in some cases, the human-robotic sex interaction would not help a user flourish or live a good life. But there may be contexts in which the interaction would enhance the user's life. Of course, the sex robot's

life should be taken into consideration. A sex robot's conception of a good life is not fully determined by its manufacturer or consumer. Manufacturers want their robots to live a life that maximizes profits for the company, a goal that would not be always ethically defensible if it were not consistent with the ability of any involved human party to live a good life or pursue well-being. Consumers may want robots to maximize pleasure—but seeking only pleasure would not be a morally defensible conception of a good life. In fact, there is no predetermined definition of a morally defensible conception of a good life, although the conception of a good life should be neither unnecessarily paternalistic, nor purely subjective. We, as a society, should ask the teleological question: What's the purpose of a sex robot and its interaction with a human?

In summation, robots interact with humans in a team-mode aggregate or ensemble, one in which robots and humans have role obligations. Robots and humans observe rites via interaction. By doing so, each entity treats the other decently. To make rites sacred (or ethically successful), participants of the rites owe role obligations to one another. The role obligation of each must be organically consistent with the purpose of the team, which can be stated generically: to achieve its context-dependent communal goal that necessarily includes a flourishing life for human and robot simultaneously.

We have been assuming that a robot can be a participant in Confucian rites. One may doubt that assumption because one might deny that a robot can understand the moral and social meaning of social interactions, that a robot can respect the moral value and standing of others, that it can have and express authentic and appropriate affective attitudes toward others. In response to this doubt, we acknowledge that given the current state of technology, there is no reason to think that robots have the mental and emotional capacity to participate in Confucian rites. But we maintain that, on at least one leading account of the mind, developments in technology will produce the needed mental and emotional capacities that are close enough. How is this possible? The answer depends on which theory of mind one embraces, functionalist or

phenomenological. On the functionalist account, the answer is easy, but on the phenomenological account, the answer is both hard and speculative. Accordingly, although we believe that we have a compelling answer on the functionalist account, which is our largest aspiration this essay, we concede that our answer on the phenomenological account is more exploratory than conclusive.

The functionalist maintains that the identity of a psychological state can be understood “functionally,” purely in terms of the role that the state plays in interacting with other psychological states, external stimuli, and behavior.²² As long as robots develop in ways that render their behavioral and inferential life sufficiently similar to that of humans, there can be no reason to deny that robots have the capacity to engage in Confucian rites, on a functionalist account, because there would be nothing in the life of humans that distinguished us from robots. On a functionalist account, robots can make the same inferences in response to the same stimuli as the rest of us, and they can follow these inferences with the same outputs that we produce.

Although functionalism is now a widely held view among philosophers of mind, it is not the only view. Many who reject functionalism maintain that human mental and emotional life has an essential conscious and subjective component that eludes functionalist analysis.²⁷ Those who doubt functionalism do not assume the perspective of conscious experience will ever be accommodated within the life of robots, in part because they do not believe that conscious experience is now well understood scientifically. That raises a hard question for us: can one reject functionalism while consistently believing that it is logically possible for a robot to participate in Confucian rites? We answer affirmatively. Human experience with sacred places confirms our answer. Consider the role of mountains in Japanese culture. They are respected. It is widely regarded as wrong to treat mountains instrumentally, for example, to develop them for commercial use. Instead, the Japanese regard it as important to preserve mountains for their own sake and to do what they can to see their mountains flourish. If mountains can thus have an elevated, even a sacred status for the Japanese,

we suggest that robots may have such a status for us, even if they don't have consciousness. So, it is possible to treat robots as having the elevated status required for them to participate in Confucian rites. The question lingers: why treat them so? Here we believe that our answer will not satisfy all. Still, we maintain that to the extent that we make robots in our image,^f if we don't treat them respectfully, as creatures capable of participating in rites, we degrade ourselves. Ritual is the way to avoid degrading them and, so, a way to respect ourselves. Now a phenomenological skeptic might respond that unless we think that robots have consciousness, we have not successfully made robots in our image, because consciousness is our essence. The skeptic's point is powerful but not conclusive. Whether something is in our image is a matter of degree of resemblance. We believe that a sophisticated robot might come close enough. To invoke a dangerous analogy, according to many of the great religions, God made us in His image, and for that reason finds it appropriate to respectfully interact with us and care about us, but the extent to which we succeed in resembling God is quite limited.

A reasonable response is available to our critic. She may say that a robot cannot be a participant in rites because a robot has no inner feeling and therefore would be no better than what Confucius calls in the *Analects* the “village goody man,” which Confucius thought, “is a thief[and the ruin] of virtue.” We suggest that the critic's response is inconclusive. A satisfying answer to the critic would include an adequate analysis of “inner feeling,” a controversial matter that is beyond our scope here. A functionalist, of course, may contend that a robot can potentially have “inner feeling” just as much as the rest of us can. We do not rely on functionalism to respond to our critic, however. Instead, we agree with our critic that the “village goody man” is a morally troubling character. The trouble with him, however, is not simply that he lacks inner feeling, but instead that he is hypocritical by falsely representing himself as having inner feeling. Our

^f Non-humanoid robots are made in our image, too, to the extent that they are powered by AI, which is, by definition, an attempt to imitate human intelligence.

robots need not do that. Still, they can resemble us to a large enough degree, as we have argued, that they may count as legitimate rites participants. No doubt our critic will deny that the degree of resemblance suffices for rites participation. That is an issue about which we respectfully disagree.^g

A major positive of the rites perspective is that it views the human-computer interaction from a team mode. In contrast, the rights perspective views the interaction as adversarial. First, the rites perspective descriptively better captures the human-computer interaction than the rights perspective, which assumes that robots are separate entities that exist individually, because the realistic view is that robots behave almost always in relationship with humans. Think about social robots in nursing homes, manufacturing robots with human workers, or even military robots working with soldiers. The team-mode view is normatively better as well. The rights view stipulates that humans and robots are adversaries who compete. Accordingly, robots' rights potentially conflict with those of humans, and the conflict must be adjudicated. This view begets the risk or the fantasy that in the future robots and humans will be embroiled in a perpetual war.

One might say that the language of rights makes only a conceptual, not substantive change. But the rights approach leads to a team featuring adversarial individuals. For example, consider a scenario where your spouse asks you to wash the dishes, but you decline, stating that you have a right to not wash the dishes. Your spouse hopes that your rationale is not a right, since you could use the same rationale when dealing with a total stranger. Through the motivating procedure, your spouse becomes alienated from the personal, special, and intimate relationship she has (had) with you. Bernard Williams memorably coins the wrongness with this type



THE ANALECTS

**Learning
without thinking
is useless.
Thinking
without learning
is dangerous.**



of rationale as being “one thought too many.”³⁸ Claiming a right depersonalizes a personal relationship, leading to a detached stranger relationship and, in this case, risks offending your spouse's affective view of her special position towards you. You could perhaps claim a right, with a detached motivation, to a non-intimate agent—thereby not necessarily degrading any intimate spirit—since you do not share such a spirit with the agent. But that would not be an appropriate motivating thought in a scenario with your spouse.

Rights talk could lead the human-robot interaction to the “one thought too many” scenario, making the two parties unnecessarily adversarial. It is not just a conceptual change, but also a motivational and behavioral one. A new theory in natural sciences (for example, physics) never actually changes reality, but a new theory or conception in social sciences or humanities can change how humans behave through self-fulfilling patterns.¹¹ For instance, learning modern economics—a study of how to optimally maximize self-interest and allocate property rights—can lead to more noncooperative behavior.¹³ Maximizing self-interest is not in itself wrong, but if granting rights to robots encourages their interaction with humans to be framed to maximize robots' interests, that is a problem. So, in terms of how to conceptualize the interaction between humans and robots, the rites perspective better addresses the risk that robots will ultimately dictate to, or fight with, humans. It can be argued that the Confucian account sketched out above is nothing more than a means-end relationship—that is, “instrumentalization”⁹—a dominant (Western) understanding of technology. If a robot functions as a means for a human to achieve her end goal, it makes little sense to respect the moral status of robots. That is, there is an inherent hierarchy in the Confucian framework that only humans have authority, while a life without authority is inconsistent with the moral status of robots.

In the Confucian rites view, however, robots can have authority and can be hierarchically superior over humans. Confucian authors (including Confucius, *Mencius*, and Xunzi) understand that authority is—and should be—created, maintained, and lost in hierarchy pri-

^g We find our view of Confucian moral psychology differs from the critic's view that inner feeling is crucial in Confucian morality and is distinctive from the physical conditions of the body and its interaction with the world.^{32,33} In our view, for Confucianism, mind is embodied in the sense that much of cognition resides in bodily movements as well as brain³¹ and the embodied cognition model well fits the interpretation of Confucian rites that we endorse.^{12,28}

marily to the extent that it earns, maintains, and loses its ethical legitimacy. This ethicized view drastically contrasts with the conventional (but mistaken) belief that in Confucianism authority in hierarchy derives from pedigree. The Confucian principle of authority submits that it must be created, maintained, and ultimately replaced based on fundamental qualities or levels of worthiness; the Confucian conception of worthiness (or merit) is understood primarily through how competent and committed the leader is to creating value(s) for all parties involved (those who have authority relationships with the leader of this hierarchy).¹⁹

Back to robots. If, in a robot-human interaction, the robot and not the human is the worthy participant because the robot, and not the human, is competent and committed to creating value(s) for all parties involved (including humans), then the robot has authority. This is not just a theoretical possibility. Think about a robot in an Amazon warehouse that exercises its algorithmic authority over workers who defer to its management of their movement of packages. The robot is competent and committed to achieving the purpose of the interaction (including enhancing operational efficiency). In this interaction, the robot's obligation is to properly exercise authority, whereas the humans' is to properly follow such authority. Of course, when the robot's authority becomes illegitimate, humans must be encouraged to communicate objections to the robot. Simultaneously, when humans have authority, robots must be designed to communicate their objections to humans.⁴⁰

The Confucian model does not always work. It is a moral system that functions for team-like, role-dependent communities (such as families or clan societies). Outside of such circumstances, we should retreat to the notion of individual rights. Jiwei Ci posited:

"[Confucians] do not know how to relate to others except on the basis of family and kinship ties ... As a result, those who have absorbed the Confucian concept of human relations would be socially and ethically at sea if they were to enter relations with strangers."⁴¹

Questions abound. The Confucian rites system works only when humans and robots mutually recognize each



THE ANALECTS

**Hold faithfulness
and sincerity
as first principles.**



other as teammates. But how can they enter this team-mode in the first place? What if humans have a prejudice that robots are not their teammates?

What if robots do not consider humans as teammates? How can the two entities enter the rites view?

The Confucian answer is, once again, the concept of rite. Rites provide a socialization process during which fractured and discrete individual entities meeting as strangers come to understand that they are now part of a specific group or team. Such socialization can occur not just in human-to-human interaction. For instance, our engagement in social rituals with pets (for example, holding a funeral) implies that the animals have some sort of status. Likewise, engaging in rituals with robots uniquely enables humans to invite robots to assimilate with them. What kinds of rites are most effective to help humans become rites-bearers is an empirical matter that deserves more research.

Additionally, how can robots be directed to treat humans as teammates? Although this article cannot offer technical solutions, it can delineate a moonshot approach. AI is the imitation of human intelligence. So, we need to first identify what mechanism of human intelligence is used to make humans rites-bearers. A human brain, especially one with an unusually large neo-cortex, has a specialized capacity for social bonding that includes the capacity to discern team activities involving group goals from those activities that do not possess such group goals.⁷ Once our brain perceives a team-framed situation, it immediately activates specialized cognitive faculties.³⁰ When individuals recognize that they are in a team activity, this capacity actively triggers participants' understanding and acceptance of the normative belief that they have role obligations they should fulfill in a harmonious manner so that their team can effectively achieve shared group goals.⁸ Such communal activities are usually initiated by "shared symbolic artifacts such a linguistic symbols and social institutions."³⁷ For robots to develop as rites-bearers, they must be powered by a kind of AI that is capable of imitating humans' capacity to recognize and execute team activities—and a machine can learn that ability in various ways.

Can an AI powered-robot recognize and understand the symbolic meaning of interactions? Fully answering this question leads to a technical discussion, the province of computer scientists that is not in the scope of this article. But, let us sketch the possibility of a robot that can recognize and process moral knowledge represented with symbolic language. At this moment, the most possible model would be a robot powered by a kind of a highly hybrid AI or the so-called “neuro-symbolic AI”¹⁴ or “neurocompositional computing.”³⁵ Such attempts combine two competing theories of mind: connectionism and computationalism. According to connectionism, the human mind can be replicated by complex artificial neural nets. Machine learning is one of the most popular realizations of connectionism. Its central advantage is its ability to recognize deep and hidden associations in training data sets, particularly in an end-to-end block box model like Deep Learning. In contrast, computationalism says that the human mind works in accordance with abstract symbol-and-rule mechanisms. Symbolic AI is a technical realization of computationalism; it is also called “good old-fashioned AI” (GOFAI). To further clarify the differences, let us use Dual Process Theory, a well-known psychological concept, according to which the human mind generally entails two different processes: System 1 and System 2.¹⁸ Neural systems are structurally parallel to System 1 because they are associative, correlational, intuitive, opaque, and fast. Symbolic AI, on the other hand, is more akin to System 2, for its processes are slow, reason-responsive and counterfactual. Similarly, Paul Smolensky once argued—with his influential Proper Treatment of Connectionism (PTC) thesis—that the human mind has two distinct functions that can be broken down to “cultural knowledge” (for example, knowledge represented with symbols and logic-following rules) and “individual knowledge” (for example, intuition, emotion, perception) and that connectionist systems relate to the latter.³⁴ Thus, to fully imitate the human mind, connectionism should embrace symbolic AI. Connectionist models such as machine learning dominate the field. But symbolic AI

is essential for a robot to properly understand role-dependent obligations, represented as a set of generalizable rules.²⁰ So, we won’t see robots as rites-bearers until we see highly neuro-symbolic robots.

Conclusion

People are worried about the risks of granting rights to robots. In this article, we have discussed a solution to address this tension, maintaining that granting rights is not the only way to treat the moral status of robots. Drawing upon Confucianism, we discussed how the distinctive concept of a moral agent as a rites-bearer, not as a rights-bearer, could be applied to robots. The Confucian alternative is superior to the robot-rights perspective because, rather than being adversarial, it is team-encouraging—unlike the concept of rights, which is inherently adversarial. Directly discussing legal issues is beyond scope of this article, but the defended view has at least one clear legal implication: Assign role obligations to robots, but do not grant them rights. It’s time to rethink robot rights. 

References

- Ames, R. Rites as rights: The Confucian alternative. *Human Rights and the World's Religions*. Routledge, 1988, 150–169.
- Asaro, P.M. What should we want from a robot ethic? *The Intern. Rev. Information Ethics* 6 (2006), 9–16.
- Chopra, S. and White, L.F. A legal theory for autonomous artificial agents. University of Michigan Press, 2011.
- Ci, J. The Confucian relational concept of the person and its modern predicament. *Kennedy Institute of Ethics J.* 9, 4 (1999), 325–346.
- Confucius. *Confucius: The Analects*. Oxford University Press, 2000.
- Danaher, J. Welcoming robots into the moral circle: A defence of ethical behaviourism. *Science and Engineering Ethics* 26, 4 (2020), 2023–2049.
- Dunbar, R. The social brain: Mind, language, and society in evolutionary perspective. *Annual Review of Anthropology* 32, 1 (2003), 163–181.
- Echterhoff, G., Higgins, E., and Levine, J. Shared reality: Experiencing commonality with others’ inner states about the world. *Perspectives on Psychological Science* 4, 5 (2009), 496–521.
- Feenberg, A., et al. *Critical Theory of Technology*, vol. 5. Oxford University Press, 1991.
- Feinberg, J. The nature and value of rights. *J. Value Inquiry* 4, 4 (1970), 243–260.
- Ferraro, F., Pfeffer, J., and Sutton, R. Economics language and assumptions: How theories can become self-fulfilling. *Academy of Management Review* 30, 1 (2005), 8–24.
- Fingarette, H. *Confucius: The Secular as Sacred*. Waverland Press, 1972.
- Frank, R., Gilovich, T., and Regan, D. Does studying economics inhibit cooperation? *J. Economic Perspectives* 7, 2 (1993), 159–171.
- Garcez, A. and Lamb, L. Neurosymbolic AI: The 3rd wave. 2020; *arXiv:2012.05876*.
- Gunkel, D. The other question: Can and should robots have rights? *Ethics and Info. Tech.* 20, 2 (2018), 87–99.
- Gunkel, D. *Robot Rights*. MIT Press, Cambridge, MA, USA, 2018.
- Thara, C. Are individual rights necessary? A Confucian perspective. *Confucian Ethics: A Comparative Study of Self, Autonomy, and Community* (2004), 11–30.
- Kahneman, D. *Thinking, Fast and Slow*. Macmillan, 2011.
- Kennedy, J., Kim, T., and Strudler, A. Hierarchies and dignity: A Confucian communitarian approach. *Business Ethics Q.* 26, 4 (2016), 479–502.
- Kim, T., Hooker, J., and Donaldson, T. Taking principles seriously: A hybrid approach to value alignment. *J. AI Research* 70 (2021), 871–890.
- Kim, T. and Strudler, A. Workplace civility: A Confucian approach. *Business Ethics Q.* 22, 3 (2012), 557–577.
- Levin, J. Functionalism. *The Stanford Encyclopedia of Philosophy*, Winter 2021. E.N. Zalta, ed. Metaphysics Research Lab, Stanford University, 2021.
- Mencius. *Mengzi: With Selections from Traditional Commentaries*. Hackett Publishing, 2008.
- Neely, E. Machines and the moral community. *Philosophy & Tech.* 27, 1 (2014), 97–111.
- Neville, R. *Boston Confucianism: Portable Tradition in the Late-Modern World*. SUNY Press, 2000.
- Pentland, B., Haerem, T., and Hillison, D. The (n) everchanging world: Stability and change in organizational routines. *Organization Science* 22, 6 (2011), 1369–1383.
- Robinson, H. *From the Knowledge Argument to Mental Substance: Resurrecting the Mind*. Cambridge University Press, 2016.
- Rosemont Jr, H. Against individualism: A Confucian rethinking of the foundations of morality, politics, family, and religion. Lexington Books, 2015.
- Schwitzgebel, E. and Garza, M. A defense of the rights of artificial intelligences. *Midwest Studies in Philosophy* 39 (2015), 98–119.
- Sebanz, N., Bekkering, H., and Knoblich, G. Joint action: Bodies and minds moving together. *Trends in Cognitive Sciences* 10, 2 (2006), 70–76.
- Seok, B. *Embodied Moral Psychology and Confucian Philosophy*. Lexington Books, 2012.
- Slingerland, E. Cognitive science and religious thought: the case of psychological interiority in the *Analects*. *Mental Culture: Classical Social Theory and the Cognitive Science of Religion*. Acumen Publishing, Stocksfield, U.K., 2013, 197–212.
- Slingerland, E. *Mind and Body in Early China: Beyond Orientalism and the Myth of Holism*. Oxford University Press, USA, 2018.
- Smolensky, P. On the proper treatment of connectionism. *Behavioral and Brain Sciences* 11, 1 (1988), 1–23.
- Smolensky, P., McCoy, R., Fernandez, R., Goldrick, M., and Gao, J. Neurocompositional computing: From the Central Paradox of Cognition to a new generation of AI systems. 2022; *abs/2205.01128*.
- Sparrow, R. Can machines be people? Reflections on the Turing triage test. *Robot Ethics: The Ethical and Social Implications of Robotics*, (2011), 301.
- Tomasello, M., Carpenter, M., Call, J., Behne, T., and Moll, H. Understanding and sharing intentions: The origins of cultural cognition. *Behavioral and Brain Sciences* 28, 5 (2005), 675–691.
- Williams, B. *Moral Luck: Philosophical Papers 1973–1980*. Cambridge University Press, 1981.
- Wong, D. Chinese ethics. *The Stanford Encyclopedia of Philosophy*, Summer 2021. E.N. Zalta, ed. Metaphysics Research Lab, Stanford University.
- Zhu, Q., Williams, T., Jackson, B., and Wen, R. Blame-laden moral rebukes and the morally competent robot: A Confucian ethical perspective. *Sci. Eng. Ethics* 26, 5 (Oct. 2020), 2511–2526.

Tae Wan Kim is an associate professor of business ethics at Carnegie Mellon University, Pittsburgh, PA, USA.

Alan Strudler is a professor of legal studies and business ethics in the Wharton School at the University of Pennsylvania, Philadelphia, PA, USA.

Copyright held by authors/owners.

 This work is licensed under a <http://creativecommons.org/licenses/by/4.0/>



Watch the authors discuss this work in the exclusive *Communications* video. <https://cacm.acm.org/videos/robots-rights-or-rites>

Examining the implicit bias in training neural networks using gradient-based methods.

BY GAL VARDI

On the Implicit Bias in Deep-Learning Algorithms

DEEP LEARNING HAS been highly successful in recent years and has led to dramatic improvements in multiple domains. Deep-learning algorithms often *generalize* quite well in practice, namely, given access to labeled training data, they return neural networks that correctly label unobserved test data. However, despite much research, our theoretical understanding of generalization in deep learning is still limited.

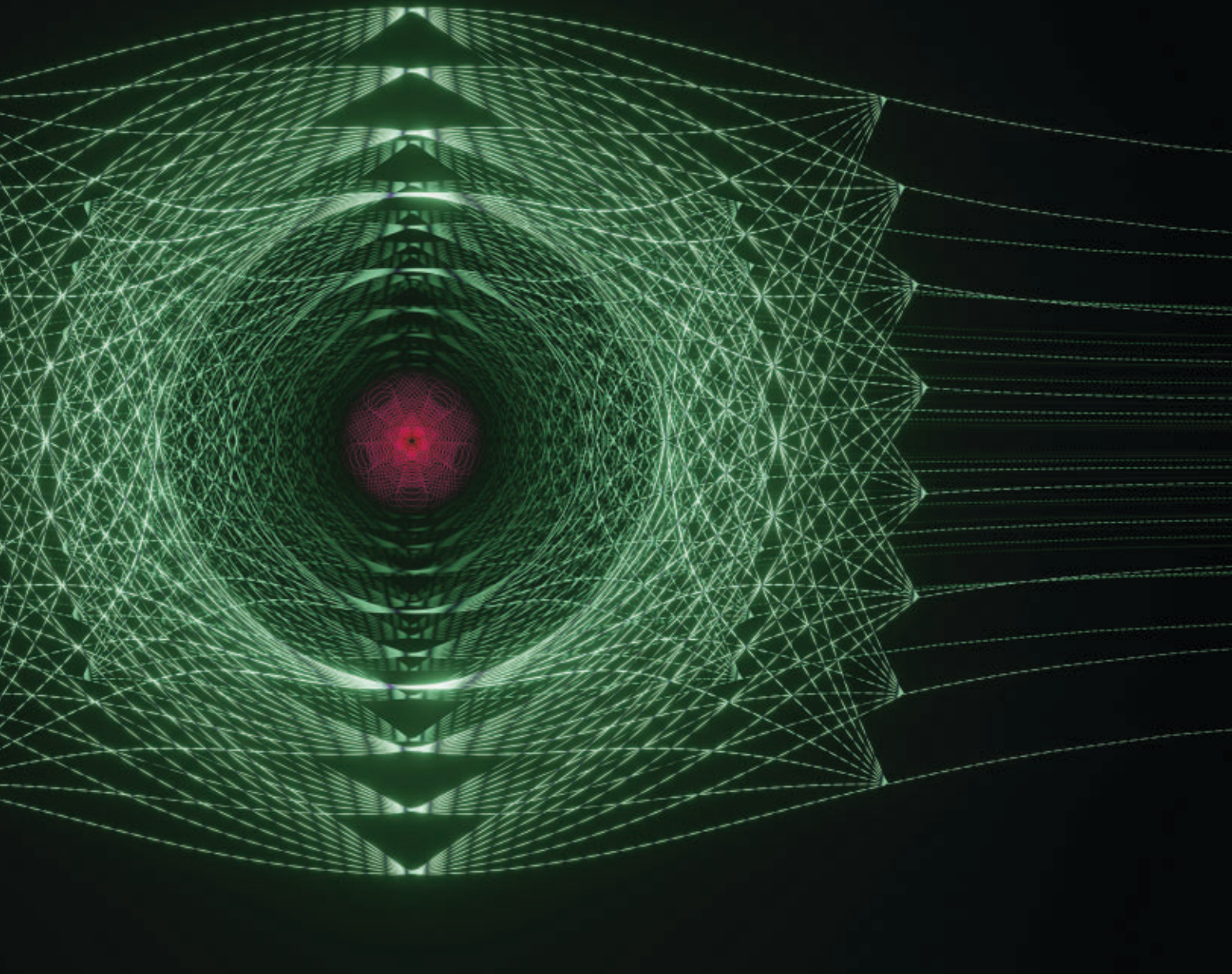
Neural networks used in practice often have far more learnable parameters than training examples. In such *overparameterized* settings, one might expect *overfitting* to occur, that is, the learned network might perform well on the training dataset and perform poorly on test data. Indeed, in overparameterized settings, there are many solutions that perform well on the training data, but most of them do not generalize well. Surprisingly, it seems

that gradient-based deep-learning algorithms^a prefer the solutions that generalize well.⁴⁰

Decades of research in learning theory suggest that in order to avoid overfitting one should use a model which is “not more expressive than necessary.” Namely, the model should be able to perform well on the training data but should be as “simple” as possible. This idea goes back to the *Occam’s Razor* philosophical principle, which argues that we should

^a Neural networks are trained using gradient-based algorithms, where the network’s parameters are randomly initialized, and then adjusted in many stages in order to fit the training dataset by using information based on the gradient of a loss function w.r.t. the network parameters.





prefer simple explanations over complicated ones. For example, in Figure 1 the data points are labeled according to a degree-3 polynomial plus a small random noise, and we fit it with a degree-3 polynomial (green) and with a degree-8 polynomial (red). The degree-8 polynomial achieves better accuracy on the training data, but it overfits and will not generalize well.

Simplicity in neural networks may stem from having a small number of parameters (which is often not the case in modern deep learning), but may also be achieved by adding a *regularization term* during the training, which encourages networks that minimize a certain measure of complexity. For example, we may add a regularizer that penalizes models where the Euclidean

norm of the parameters (viewed as a vector) is large. However, in practice, neural networks seem to generalize well even when trained without such an explicit regularization,⁴⁰ and hence the success of deep learning cannot be attributed to explicit regularization. Therefore, it is believed that gradient-based algorithms induce an *implicit bias* (or *implicit regularization*)²⁸ which prefers solutions that generalize well, and characterizing this bias has been a subject of extensive research.

In this review article, we discuss the implicit bias in training neural networks using gradient-based methods. The literature on this subject has rapidly expanded in recent years, and we aim to provide a high-level overview of some fundamental results. This article

is not a comprehensive survey, and there are certainly important results which are not discussed here.

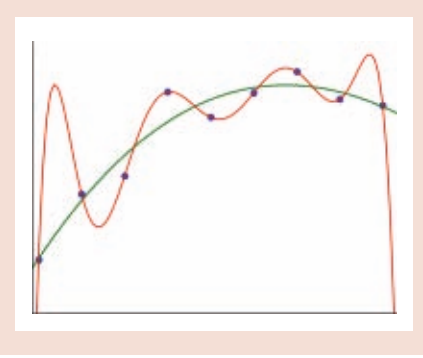
The Double-Descent Phenomenon

An important implication of the implicit bias in deep learning is the

» key insights

- Deep-learning algorithms exhibit remarkable performance in practice, but it is not well-understood why they are able to generalize despite having more parameters than training examples.
- Exploring generalization in deep learning is an intriguing puzzle with both theoretical and practical implications.
- It is believed that implicit bias is a key factor in the ability of deep-learning algorithms to generalize.

Figure 1. Fitting training data with a degree-3 polynomial (green) and degree-8 polynomial (red). The latter achieves better accuracy on the training data but it overfits.



double-descent phenomenon, observed by Belkin et al.⁶ As we already discussed, conventional wisdom in machine learning suggests that the number of parameters in the neural network should not be too large in order to avoid overfitting (when training without explicit regularization). Also, it should not be too small, in which case the model is not expressive enough and hence it performs poorly even on the training data, a situation called *underfitting*. This classical thinking can be captured by the U-shaped risk (that is, error or loss) curve from Figure 2A. Thus, as we increase the number of parameters, the training risk decreases, and the test risk initially decreases and then increases. Hence, there is a “sweet spot” where the test risk is minimized. This classical U-shaped curve suggests that we should not use a model that perfectly fits the training data.

However, it turns out that the U-shaped curve only provides a partial picture.

If we continue to increase the number of parameters in the model after the training set is already perfectly labeled, then there might be a second descent in the test risk (hence the name “double descent”). As can be seen in Figure 2B, by increasing the number of parameters beyond what is required to perfectly fit the training set, the generalization improves. Hence, in contrast to the classical approach which seeks a sweet spot, we may achieve generalization by using overparameterized neural networks. Modern deep learning indeed relies on using highly overparameterized networks.

The double-descent phenomenon is believed to be a consequence of the implicit bias in deep-learning algorithms. When using overparameterized models, gradient methods converge to networks that generalize well by implicitly minimizing a certain measure of the network’s complexity. As we will see next, characterizing this complexity measure in different settings is a challenging puzzle.

Implicit Bias in Classification

We start with the implicit bias in classification tasks, namely, where the labels are in $\{-1, 1\}$. Neural networks are trained in practice using the *gradient-descent* algorithm and its variants, such as *stochastic gradient descent* (SGD). In gradient descent, we start from some random initialization θ_0 of the network’s parameters, and then in each iteration $t \geq 1$, we update $\theta_t = \theta_{t-1} - \eta \nabla \mathcal{L}(\theta_{t-1})$, where $\eta > 0$ is the step size and $\mathcal{L}$ is some *empirical loss* that we wish to minimize. Here, we will focus on *gradient flow*, which is a continuous version of

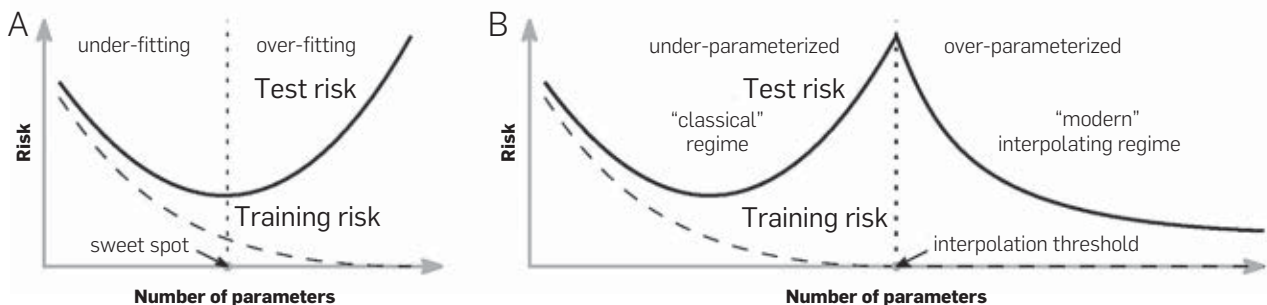
gradient descent. That is, we train the parameters θ of the considered neural network, such that for all time $t \geq 0$ we have $\frac{d\theta(t)}{dt} = -\nabla \mathcal{L}(\theta(t))$, where $\mathcal{L}$ is the empirical loss, and $\theta(0)$ is some random initialization.^b We focus on the *logistic loss function* (a.k.a. *binary cross entropy*): For a training dataset $\{(x_i, y_i)\}_{i=1}^n \subseteq \mathbb{R}^d \times \{-1, 1\}$ and a neural network $\mathcal{N}_\theta: \mathbb{R}^d \rightarrow \mathbb{R}$ parameterized by θ , the empirical loss is defined by $\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^n \log(1 + e^{-y_i \cdot \mathcal{N}_\theta(x_i)})$. We note that gradient flow corresponds to gradient descent with an infinitesimally small step size, and many implicit-bias results are shown for gradient flow since it is often easier to analyze.

Logistic regression. Logistic regression is perhaps the simplest classification setting where the implicit bias of neural networks can be considered. It corresponds to a neural network with a single neuron that does not have an activation function. That is, the model here is a linear predictor $x \mapsto \mathbf{w}^\top x$, where $\mathbf{w} \in \mathbb{R}^d$ are the learned parameters.

We assume that the data is *linearly separable*, that is, there exists some $\hat{\mathbf{w}} \in \mathbb{R}^d$ such that for all $i \in \{1, \dots, n\}$, we have $y_i \cdot \hat{\mathbf{w}}^\top x_i > 0$. Note that $\mathcal{L}(\mathbf{w})$ is strictly positive for all $\mathbf{w}$, but it may tend to zero as $\|\mathbf{w}\|_2$ tends to infinity. For example, for $\alpha \in \mathbb{R}$ and $\mathbf{w} = \alpha \hat{\mathbf{w}}$ (where $\hat{\mathbf{w}}$ is the vector defined earlier) we have $\lim_{\alpha \rightarrow \infty} \mathcal{L}(\mathbf{w}) = 0$. However, there might be infinitely many directions $\hat{\mathbf{w}}$ that satisfy this condition. Interestingly, Soudry et al.³³ showed that gradient

^b If $\mathcal{L}$ is non-differentiable, for example, in ReLU networks, then we use the *Clarke subdifferential*, which is a generalization of the derivative for non-differentiable functions.

Figure 2. The double-descent phenomenon. Dashed lines denote the training risk and solid lines denote the test risk. (A) The classical U-shaped curve; and (B) The double-descent curve, extends the U-shaped curve with a second descent. Beyond the “interpolation threshold” the model fits the training data perfectly, but the test risk decreases. The figure is from Belkin et al.⁶



flow converges in the direction to the ℓ_2 maximum-margin predictor. That is, $\tilde{\mathbf{w}} = \lim_{t \rightarrow \infty} \frac{\mathbf{w}(t)}{\|\mathbf{w}(t)\|_2}$ exists, and $\tilde{\mathbf{w}}$ points in the direction of the maximum-margin predictor, defined by

$$\operatorname{argmin}_{\mathbf{w} \in \mathbb{R}^d} \|\mathbf{w}\|_2 \quad \text{s.t.} \quad \forall i \in \{1, \dots, n\} \quad y_i \cdot \mathbf{w}^\top \mathbf{x}_i \geq 1.$$

The characterization of the implicit bias in logistic regression can be used to explain generalization. Consider the case where $d > n$, thus, the input dimension is larger than the size of the dataset, and hence there are infinitely many vectors $\mathbf{w}$ with $\|\mathbf{w}\|_2 = 1$ such that for all $i \in \{1, \dots, n\}$, we have $y_i \cdot \mathbf{w}^\top \mathbf{x}_i > 0$, that is, there are infinitely many directions that correctly label the training data. Some of the directions which fit the training data generalize well, and others do not. The result guarantees that gradient flow converges to the maximum-margin direction. Consequently, we can explain generalization by using standard margin-based generalization bounds (Shalev-Shwartz and Ben-David³²), which imply that predictors achieving large margins generalize well.

Linear networks. We turn to deep neural networks where the activation is the identity function. Such neural networks are called *linear networks*. These networks compute linear functions, but the network architectures induce significantly different dynamics of gradient flow compared to the case of linear predictors $\mathbf{x} \mapsto \mathbf{w}^\top \mathbf{x}$ that we already discussed. Hence, the implicit bias in linear networks has been extensively studied, as a first step toward understanding implicit bias in deep nonlinear networks. It turns out that understanding implicit bias in linear networks is highly non-trivial, and its study reveals valuable insights.

A linear *fully-connected* network $\mathcal{N}: \mathbb{R}^d \rightarrow \mathbb{R}$ of depth k computes a function $\mathcal{N}(\mathbf{x}) = W_k \dots W_1 \mathbf{x}$, where $W_1, \dots, W_k$ are weight matrices (where W_k is a row vector). The trainable parameters are a collection $\theta = [W_1, \dots, W_k]$ of the weight matrices. By Ji and Telgarsky^{17,c} if gradient flow converges to zero loss, namely, $\lim_{t \rightarrow \infty} \mathcal{L}(\theta(t)) = 0$, then the vector $W_k \cdot \dots \cdot W_1$ converges in direction

to the ℓ_2 maximum-margin predictor. That is, although the dynamics of gradient flow in the case of a deep fully-connected linear network is significantly different compared to the case of a linear predictor, in both cases gradient flow is biased toward the ℓ_2 maximum-margin predictor. We also note that by Ji and Telgarsky,¹⁶ the weight matrices $W_1, \dots, W_{k-1}$ converge to matrices of rank 1. Thus, gradient flow on fully-connected linear networks is also biased toward rank minimization of the weight matrices.

Once we consider linear networks which are not fully connected, gradient flow no longer maximizes the ℓ_2 margin. For example, Gunasekar et al.¹³ showed that in *linear diagonal networks* (that is, where the weight matrices $W_1, \dots, W_{k-1}$ are diagonal) gradient flow encourages predictors that maximize the margin w.r.t. the $\|\cdot\|_{2/k}$ quasi-norm. As a result, diagonal networks are biased toward sparse linear predictors. In *linear convolutional networks*, they proved bias toward the sparsity of the linear predictors in the frequency domain (see also Yun et al.³⁹).

Homogeneous neural networks. Neural networks of practical interest have nonlinear activations and compute nonlinear predictors. The notion of margin maximization in nonlinear predictors is generally not well-defined. Nevertheless, it has been established that for certain neural networks, gradient flow maximizes the margin in *parameter space*. Thus, while in linear networks, we considered margin maximization in *predictor space* (a.k.a. *function space*), here we consider margin maximization w.r.t. the network's parameters.

Consider a neural network with parameters θ , denoted by $\mathcal{N}_\theta: \mathbb{R}^d \rightarrow \mathbb{R}$. We say that $\mathcal{N}_\theta$ is *homogeneous* if there exists $L > 0$ such that for every $\alpha > 0$ and $\theta, \mathbf{x}$ we have $\mathcal{N}_{\alpha\theta}(\mathbf{x}) = \alpha^L \mathcal{N}_\theta(\mathbf{x})$. Here we think about θ as a vector obtained by concatenating all the parameters. That is, in homogeneous networks, scaling the parameters by any factor $\alpha > 0$ scales the predictions by α^L . We note that a feedforward neural network with the ReLU activation (namely, $z \mapsto \max\{0, z\}$) is homogeneous if it does not contain skip-connections (that is, residual connections), and does not have bias terms, except maybe for the

first hidden layer. Also, we note that a homogeneous network might include convolutional layers. In papers by Lyu and Li²¹ and by Ji and Telgarsky,^{17,d} it was shown that if gradient flow on homogeneous networks reaches a sufficiently small loss (at some time t_0), then as $t \rightarrow \infty$ it converges to zero loss, the parameters converge in direction, and $\tilde{\theta} = \lim_{t \rightarrow \infty} \frac{\theta(t)}{\|\theta(t)\|_2}$ exists, $\tilde{\theta}$ and is biased toward margin maximization in the following sense. Consider the following margin maximization problem in parameter space:

$$\min_{\tilde{\theta}} \frac{1}{2} \|\tilde{\theta}\|_2^2 \quad \text{s.t.} \quad \forall i \in \{1, \dots, n\} \quad y_i \mathcal{N}_{\tilde{\theta}}(\mathbf{x}_i) \geq 1$$

Then, $\tilde{\theta}$ points at the direction of a first-order stationary point of the optimization problem, which is also called *Karush-Kuhn-Tucker (KKT) point*. The KKT approach allows inequality constraints and is a generalization of the method of *Lagrange multipliers*, which allows only equality constraints.

A KKT point satisfies several conditions (called *KKT conditions*), and it was proved in Lyu and Li²¹ that in the case of homogeneous neural networks, these conditions are necessary for optimality. However, they are not sufficient even for local optimality (Vardi et al.³⁵). Thus, a KKT point may not be an actual optimum of the maximum-margin problem. It is analogous to showing that some unconstrained optimization problem converges to a point where the gradient is zero, without proving that it is a global/local minimum. Intuitively, convergence to a KKT point of the maximum-margin problem implies a certain bias toward margin maximization in parameter space, although it does not guarantee convergence to a maximum-margin solution.

As we already discussed, in linear classifiers and linear neural networks, margin maximization (in predictor space) can explain generalization using well-known margin-based generalization bounds. Can margin maximization in parameter space explain generalization in nonlinear neural networks? In recent years several works showed margin-based generalization bounds for neural networks (for example, Bartlett et al.⁵ and Golowich et al.¹¹). Hence,

c Similar results under stronger assumptions were previously established in Gunasekar et al.,¹³ Ji and Telgarsky.¹⁶

d A similar result under stronger assumptions was previously established in Nacson et al.²⁶

generalization in neural networks might be established by combining these results with results on the implicit bias toward margin maximization in parameter space. On the flip side, we note that it is still unclear how tight the known margin-based generalization bounds for neural networks are, and whether the sample complexity implied by such results (that is, the required size of the training dataset) may be small enough to capture the situation in practice.

Several results on implicit bias were shown for some specific cases of nonlinear homogeneous networks. For example: Chizat and Bach⁸ showed bias toward margin maximization w.r.t. a certain function norm (known as the variation norm) in infinite-width two-layer homogeneous networks; Lyu et al.²² proved margin maximization in two-layer Leaky-ReLU networks trained on linearly separable and symmetric data, and Sarussi et al.³¹ proved convergence to a linear classifier in two-layer Leaky-ReLU networks under different assumptions; Safran et al.³⁰ showed bias toward minimizing the number of linear regions in univariate two-layer ReLU networks.

Finally, the implicit bias in *non-homogeneous neural networks* (such as ResNets) is currently not well-understood.^e Improving our knowledge of non-homogeneous networks is an important challenge on the path toward understanding implicit bias in deep learning.

Extensions. For simplicity, we considered so far only gradient flow in binary classification. We note that many of the results here can also be extended to other gradient methods (such as gradient descent, steepest descent, and SGD), and to multiclass classification with the cross-entropy loss. The results can also be extended to various loss functions. Furthermore, we focused on an asymptotic analysis, that is, where the time t tends to infinity, but the analysis can be extended to the case where gradient descent is trained for a finite number of iterations. We discuss all these important extensions in an online appendix.

Additional aspects of the implicit bias, which apply both to classification and regression, are discussed later.

Implicit Bias in Regression

We turn to consider the implicit bias of gradient methods in regression tasks, namely, where the labels are in $\mathbb{R}$. We focus on gradient flow and on the *square-loss* function. Thus, we have $\frac{d\theta(t)}{dt} = -\nabla \mathcal{L}(\theta(t))$, where the empirical loss $\mathcal{L}(\theta)$ is defined such that given a training dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n \subseteq \mathbb{R}^d \times \mathbb{R}$ and a neural network $\mathcal{N}_\theta: \mathbb{R}^d \rightarrow \mathbb{R}$, we have $\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^n (\mathcal{N}_\theta(\mathbf{x}_i) - y_i)^2$. In an overparameterized setting, there might be many possible choices of θ such that $\mathcal{L}(\theta) = 0$, thus there are many global minima. Ideally, we want to find some implicit regularization function $\mathcal{R}(\theta)$ such that gradient flow prefers global minima which minimize $\mathcal{R}$. That is, if gradient flow converges to some θ^* with $\mathcal{L}(\theta^*) = 0$, then we have $\theta^* \in \arg\min_\theta \mathcal{R}(\theta)$ s.t. $\mathcal{L}(\theta) = 0$.

Linear regression. We start with linear regression, which is perhaps the simplest regression setting where the implicit bias of neural networks can be considered. Here, the model is a linear predictor $\mathbf{x} \mapsto \mathbf{w}^\top \mathbf{x}$, where $\mathbf{w} \in \mathbb{R}^d$ are the learned parameters. In this case, it is not hard to show that gradient flow converges to the global minimum of $\mathcal{L}(\mathbf{w})$, which is closest to the initialization $\mathbf{w}(0)$ in ℓ_2 distance.^{12,40} Thus, the implicit regularization function is $\mathcal{R}_{\mathbf{w}(0)}(\mathbf{w}) = \|\mathbf{w} - \mathbf{w}(0)\|_2$. Minimizing this ℓ_2 norm allows us to explain generalization using standard norm-based generalization bounds.³²

We note that the results on linear regression can be extended to optimization methods other than gradient flow, such as gradient descent, SGD, and *mirror descent*.^{3,12,f}

Linear networks. A linear neural network $\mathcal{N}_\theta: \mathbb{R}^d \rightarrow \mathbb{R}$ has the identity activation function and computes a linear predictor $\mathbf{x} \mapsto \mathbf{w}^\top \mathbf{x}$, where $\mathbf{w}$ is the vector obtained by multiplying the weight matrices in $\mathcal{N}_\theta$. Hence, instead of seeking an implicit regularization function $\mathcal{R}(\theta)$, we may aim to find $\mathcal{R}(\mathbf{w})$.

Similarly to our discussion in the context of classification, the network architectures in deep linear networks induce significantly different dynamics of gradient flow compared to the case of linear regression. Hence, the implicit bias in such networks has been studied as a first step toward understanding implicit bias in deep nonlinear networks.

Exact expressions for $\mathcal{R}(\mathbf{w})$ have been obtained for linear diagonal networks and linear fully-connected networks.^{4,36,39} The expressions for $\mathcal{R}(\mathbf{w})$ are rather complicated, and depend on the initialization scale, that is, the norm of $\mathbf{w}(0)$, and the initialization “shape,” namely, the relative magnitudes of different weights and layers in the network. Here, we discuss a few special cases.

Recall that diagonal networks are neural networks where the weight matrices are diagonal. The regularization function $\mathcal{R}$ has been obtained for a few variants of diagonal networks. In Woodworth et al.³⁶ the authors considered networks of the form $\mathbf{x} \mapsto (\mathbf{u}_+ \circ \mathbf{u}_+ - \mathbf{u}_- \circ \mathbf{u}_-)^\top \mathbf{x}$, where $\mathbf{u}_+, \mathbf{u}_- \in \mathbb{R}^d$ and the operator $\circ$ denotes the Hadamard (entrywise) product. Thus, in this network, which can be viewed as a variant of a diagonal network, the weights in the two layers are shared. For an initialization $\mathbf{u}_+(0) = \mathbf{u}_-(0) = \alpha \cdot \mathbf{1}$, the scale $\alpha > 0$ of the initialization does not affect $\mathbf{w}(0) = \mathbf{u}_+(0) \circ \mathbf{u}_+(0) - \mathbf{u}_-(0) \circ \mathbf{u}_-(0) = \mathbf{0}$. However, the initialization scale does affect the implicit bias: if $\alpha \rightarrow 0$ then $\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_1$ and if $\alpha \rightarrow \infty$ (which corresponds to the so-called *kernel regime*) then $\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_2$. Thus, the implicit bias transitions between the ℓ_1 norm and the ℓ_2 norm as we increase the initialization scale. A similar result holds also for deeper networks. In two-layer “diagonal networks” with a similar structure, but where the weights are not shared, Azulay et al.⁴ showed that the implicit bias is affected by both the initialization scale and “shape.” In Yun et al.³⁹ the authors showed a transition between the ℓ_1 and ℓ_2 norms both in diagonal networks and in certain convolutional networks (in the frequency domain).

In two-layer fully-connected linear networks with infinitesimally small initialization, gradient flow minimizes the ℓ_2 norm, namely, $\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_2$.^{4,39} This result was shown also for

^e We remark that a result by Nacson et al.²⁶ suggests that for a sum of homogeneous networks of different orders (such a sum is non-homogeneous), the implicit bias may encourage solutions that discard the networks with the smallest order.

^f For mirror descent with a potential function $\psi(\mathbf{w})$, the implicit bias is given by $\mathcal{R}(\mathbf{w}) = D_\psi(\mathbf{w}, \mathbf{w}(0))$, where D_ψ is the Bregman divergence w.r.t. ψ . In particular, if we start at $\mathbf{w}(0) = \arg\min_{\mathbf{w}} \psi(\mathbf{w})$ then we have $\mathcal{R}(\mathbf{w}) = \psi(\mathbf{w})$.

deep fully-connected linear networks, under certain assumptions on the initialization.³⁹

Some additional aspects of the implicit bias in linear networks are discussed later.

Matrix factorization as a test-bed for neural networks. Toward the goal of understanding implicit bias in neural networks, much effort was directed at the *matrix factorization* (and more specifically, *matrix sensing*) problem. In this problem, we are given a set of observations about an unknown matrix $W^* \in \mathbb{R}^{d \times d}$ and we find a matrix $W = W_2 W_1$ with $W_1, W_2 \in \mathbb{R}^{d \times d}$ that is compatible with the given observations by running gradient flow. More precisely, consider the observations $\{(X_i, y_i)\}_{i=1}^n \subseteq \mathbb{R}^{d \times d} \times \mathbb{R}$ such that $y_i = \langle W^*, X_i \rangle$, where in the inner product, we view W^* and X_i as vectors (namely, $\langle W, X \rangle = \text{tr}(W^\top X)$). Then, we find $W = W_2 W_1$ using gradient flow on the objective $\mathcal{L}(W_1, W_2) = \frac{1}{n} \sum_{i=1}^n (\langle W_2 W_1, X_i \rangle - y_i)^2$. Since $W_1, W_2 \in \mathbb{R}^{d \times d}$ then the rank of W is not limited by the parameterization. However, the parameterization has a crucial effect on the implicit bias. Assuming that there are many matrices W that are compatible with the observations, the implicit bias controls which matrix gradient flow finds. A notable special case of matrix sensing is *matrix completion*. Here, every observation has the form $X_i = \mathbf{e}_{p_i} \mathbf{e}_{q_i}^\top$, where $\{\mathbf{e}_1, \dots, \mathbf{e}_d\}$ is the standard basis. Thus, each observation (X_i, y_i) reveals a single entry from the matrix W^* .

The matrix factorization problem is analogous to training a two-layer linear network. Furthermore, we may consider *deep matrix factorization* where we have $W = W_k W_{k-1} \dots W_1$, which is analogous to training a deep linear network. Therefore, this problem is considered a natural test-bed to investigate implicit bias in neural networks, and has been studied extensively in recent years (for example, Arora et al.,¹ Gunasekar et al.,¹⁴ Li et al.,^{19,20} Razin and Cohen²⁹).

Gunasekar et al.¹⁴ conjectured that in matrix factorization, the implicit bias of gradient flow starting from small initialization is given by the nuclear norm of W (a.k.a. the trace norm). That is, $\mathcal{R}(W) = \|W\|_*$. They also proved this conjecture in the restricted case where the matrices

The matrix factorization problem is analogous to training a two-layer linear network.

$\{X_i\}_{i=1}^n$ commute. The conjecture was further studied in a line of works (Arora et al.,¹ Li et al.,¹⁹ Razin and Cohen²⁹) providing positive and negative evidence, and was formally refuted by Li et al.²⁰ In Razin and Cohen²⁹ the authors showed that gradient flow in matrix completion might approach a global minimum at infinity rather than converging to a global minimum with a finite norm. This result suggests that the implicit bias in matrix factorization may not be expressible by any norm or quasi-norm. They conjectured that the implicit bias can be explained by rank minimization rather than norm minimization. Li et al.²⁰ provide theoretical and empirical evidence that gradient flow with infinitesimally small initialization in the matrix-factorization problem is mathematically equivalent to a simple heuristic rank-minimization algorithm called *Greedy Low-Rank Learning*.

Overall, although an explicit expression for a function $\mathcal{R}$ is not known in the case of matrix factorization, we can view the implicit bias as a heuristic for rank minimization. It is still unclear what the implications of these results are for more practical nonlinear neural networks.

Nonlinear networks. Once we consider nonlinear networks the situation is even less clear. For a single-neuron network, namely, $\mathbf{x} \mapsto \sigma(\mathbf{w}^\top \mathbf{x})$, where $\sigma: \mathbb{R} \rightarrow \mathbb{R}$ is strictly monotonic, gradient flow converges to the closest global minimum in ℓ_2 distance, similarly to the case of linear regression. Thus, we have $\mathcal{R}_{\mathbf{w}(0)}(\mathbf{w}) = \|\mathbf{w} - \mathbf{w}(0)\|_2$. However, if σ is the ReLU activation then the implicit bias is not expressible by any non-trivial function of $\mathbf{w}$. A bit more precisely, suppose that $\mathcal{R}: \mathbb{R}^d \rightarrow \mathbb{R}$ is such that if gradient flow starting from $\mathbf{w}(0) = \mathbf{0}$ converges to a zero-loss solution $\mathbf{w}^*$, then $\mathbf{w}^*$ is a zero-loss solution that minimizes $\mathcal{R}$. Then, such a function $\mathcal{R}$ must be constant in $\mathbb{R}^d \setminus \{\mathbf{0}\}$. Hence, the approach of precisely specifying the implicit bias of gradient flow via a regularization function is not feasible in single-neuron ReLU networks.³⁴ It suggests that this approach may not be useful for understanding implicit bias in the general case of ReLU networks.

On the positive side, in single-neuron ReLU networks, the implicit bias can be expressed approximately, within a factor of 2, by the ℓ_2 norm. Namely, let $\mathbf{w}^*$

more online

To view the online appendix visit <https://dl.acm.org/doi/10.1145/3571070>

be a zero-loss solution with a minimal ℓ_2 norm, and assume that gradient flow converges to some zero-loss solution $\mathbf{w}^{(\infty)}$, then $\|\mathbf{w}^{(\infty)}\|_2 \leq 2 \|\mathbf{w}^*\|_2$.³⁴

For two-layer Leaky-ReLU networks with a single hidden neuron, an explicit expression for $\mathcal{R}$ was obtained in Azulay et al.⁴ However, if we replace the Leaky-ReLU activation with ReLU, then the implicit bias is not expressible by any non-trivial function, similar to the case of a single-neuron ReLU network.³⁴

Overall, our understanding of implicit bias in nonlinear networks with regression losses is very limited. While in classification we have a useful characterization of the implicit bias in nonlinear homogeneous models, here we do not understand even extremely simple models. Improving our understanding of this question is a major challenge for the upcoming years.

In what follows, we will discuss some additional aspects of the implicit bias, which apply both to regression and classification.

Dynamical Stability

In the previous sections, we mostly focused on implicit bias in gradient flow, and also discussed cases where the results on gradient flow can be extended to other gradient methods. However, when considering gradient descent or SGD with a finite step size (rather than infinitesimal), the discrete nature of the algorithms and the stochasticity induce additional implicit biases that do not exist in gradient flow. These biases are crucial for understanding the behavior of gradient methods in practice, as empirical results suggest that increasing the step size and decreasing the batch size may improve generalization (Jastrzębski et al.,¹⁵ Keskar et al.¹⁸).

It is well known that gradient descent and SGD cannot stably converge to minima that are too sharp relative to the step size (see, for example, Mulayoff and Michaeli,²⁴ Mulayoff et al.,²⁵ Nacson et al.,²⁷ Wu et al.³⁷). Namely, in a stable minimum, the maximal eigenvalue of the Hessian is at most $2/\eta$, where η is the step size. As a result, when using an appropriate step size we can rule out stable convergence to certain (sharp) minima,

Our understanding of implicit bias in nonlinear networks with regression losses is very limited.

and thus encourage convergence to flat minima, which are believed to generalize better.^{18, 37} Similarly, small batch sizes in SGD also encourage flat minima.^{18, 23, 37, 38}

We note that a function might be represented with many different networks, that is, there are many sets of parameters that correspond to the same function. However, it is possible that some of these representations are sharp (in parameter space) and others are flat (Dinh et al.¹⁰). As a result, understanding the implications in the function space of the bias toward flat minima (in parameter space) is often a challenging question.

Implications of the implicit bias toward flat minima were studied in several works. For example, Mulayoff et al.²⁵ studied flat minima in univariate ReLU networks, and showed that SGD with large step size is biased toward functions whose second derivative (w.r.t. the input) has a bounded weighted ℓ_1 norm, which encourages convergence to smooth functions; Nacson et al.²⁷ showed that gradient descent with large step size on a diagonal linear network with non-centered data minimizes the ℓ_1 norm of the corresponding linear predictor, even in settings where gradient descent with small step size minimizes the ℓ_2 norm; Ma and Ying²³ studied minima stability in SGD, and showed that flat minima induce regularization of the function's gradient (w.r.t. the input data), and more generally that SGD regularizes Sobolev seminorms of the function; Mulayoff and Michaeli²⁴ studied flat minima in linear neural networks and showed bias toward solutions where the magnitudes of the layers are balanced, as well as other properties of flat solutions. We note that all these papers consider regression with the square loss.

An intriguing related phenomenon, observed by Cohen et al.,⁹ is the tendency of gradient descent to operate in a regime called *the Edge of Stability (EoS)*. They showed empirically for several architectures and tasks, and for both the cross-entropy and the square loss, that the behavior of gradient descent with step size η consists of two stages. First, the loss curvature grows until the sharpness touches the bound of $2/\eta$. Then, gradient descent enters the EoS regime,

where curvature hovers around this bound, and the train loss behaves non-monotonically, yet consistently decreases over long timescales. This phenomenon is not well understood, as traditional convergence analyses of gradient descent do not apply when the sharpness is above $2/\eta$. The study of the EoS regime may contribute to our understanding of the dynamics and implicit bias in gradient descent and SGD. It was investigated in several recent theoretical works. For example, Arora et al² showed that training with modified gradient descent or modified loss (for a large family of losses) provably enters the EoS regime, and that the loss decreases in a non-monotone manner, and Chen and Bruna⁷ studied EoS in training a two-layer single-neuron ReLU network with the square loss.

Additional Implicit Biases and Further Implications

Additional aspects of the implicit bias were studied in the literature. These aspects include the effect of label noise on the implicit bias in SGD; transforming gradient descent and SGD into gradient flow on a modified loss; balancedness of weights' magnitudes between different layers; implications of norm minimization in parameter space on the predictor in function space; as well as implicit bias in the NTK regime. We briefly discuss these aspects as well as further implications of the implicit bias other than generalization, namely, implications on adversarial robustness and on privacy in an online appendix (<https://dl.acm.org/doi/10.1145/3571070>)

Conclusion

Deep-learning algorithms exhibit remarkable performance, but it is not well-understood why they are able to generalize despite having much more parameters than training examples. Exploring generalization in deep learning is an intriguing puzzle with both theoretical and practical implications. It is believed that implicit bias is a main ingredient in the ability of deep-learning algorithms to generalize, and hence it has been widely studied.

Our understanding of implicit bias improved dramatically in recent years but is still far from satisfactory. We believe that further progress in the

study of implicit bias will eventually shed light on the mystery of generalization in deep learning.

Acknowledgments

I thank Nadav Cohen, Noam Razin, Ohad Shamir, and Daniel Soudry for valuable comments and discussions. 

References

- Arora, S., Cohen, N., Hu, W., Luo, Y. Implicit regularization in deep matrix factorization. In *Advances in Neural Information Processing Systems* (2019), Curran Associates, Inc, Red Hook, NY, 7413–7424.
- Arora, S., Li, Z., Panigrahi, A. Understanding gradient descent on edge of stability in deep learning. arXiv preprint arXiv:2205.09745 (2022).
- Azizan, N., Hassibi, B. Stochastic gradient/mirror descent: Minimax optimality and implicit regularization. In *International Conference on Learning Representations (ICLR)* (2019), OpenReview.net.
- Azulay, S., Moroshko, E., Nacson, M.S., Woodworth, B., Srebro, N., Globerson, A., et al. On the implicit bias of initialization shape: Beyond infinitesimal mirror descent. In *International Conference on Machine Learning* (2021), PMLR, 468–477.
- Bartlett, P.L., Foster, D.J., Telgarsky, M.J. Spectrally-normalized margin bounds for neural networks. *Adv. Neural Inf. Process. Syst.* 30 (2017), 6240–6249.
- Belkin, M., Hsu, D., Ma, S., Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proc. Natl. Acad. Sci.* 116, 32 (2019), 15849–15854.
- Chen, L., Bruna, J. On gradient descent convergence beyond the edge of stability. arXiv preprint arXiv:2206.04172 (2022).
- Chizat, L., Bach, F. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *Conference on Learning Theory* (2020), PMLR, 1305–1338.
- Cohen, J.M., Kaur, S., Li, Y., Zico Kolter, J., Talwalkar, A. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations (ICLR)* (2021), OpenReview.net.
- Dinh, L., Pascanu, R., Bengio, S., Bengio, Y. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning* (2017), PMLR, 1019–1028.
- Golowich, N., Rakhlin, A., Shamir, O. Size-independent sample complexity of neural networks. In *Conference on Learning Theory* (2018), PMLR, 297–299.
- Gunasekar, S., Lee, J., Soudry, D., Srebro, N. Characterizing implicit bias in terms of optimization geometry. In *International Conference on Machine Learning* (2018), PMLR, 1832–1841.
- Gunasekar, S., Lee, J.D., Soudry, D., Srebro, N. Implicit bias of gradient descent on linear convolutional networks. In *Advances in Neural Information Processing Systems* (2018), Curran Associates, Inc, Red Hook, NY, 9461–9471.
- Gunasekar, S., Woodworth, E.B., Bhojanapalli, S., Neyshabur, B., Srebro, N. Implicit regularization in matrix factorization. *Adv. Neural Inf. Process. Syst.* 30 (2017), 6151–6159.
- Jastrzebski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., et al. Three factors influencing minima in SGD. arXiv preprint arXiv:1711.04623 (2017).
- Ji, Z., Telgarsky, M. Gradient descent aligns the layers of deep linear networks. In *International Conference on Learning Representations* (2018), OpenReview.net.
- Ji, Z., Telgarsky, M. Directional convergence and alignment in deep learning. In *Advances in Neural Information Processing Systems* (2020), Curran Associates, Inc, Red Hook, NY.
- Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations (ICLR)* (2017), OpenReview.net.
- Li, Y., Ma, T., Zhang, H. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. In *Conference on Learning Theory* (2018), PMLR, 2–47.
- Li, Z., Luo, Y., Lyu, K. Towards Resolving the Implicit Bias of Gradient Descent for Matrix Factorization: Greedy Low-Rank Learning. In *International Conference on Learning Representations* (2020), OpenReview.net.
- Lyu, K., Li, J. Gradient descent maximizes the margin of homogeneous neural networks. In *International Conference on Learning Representations* (2019).
- Lyu, K., Li, Z., Wang, R., Arora, S. Gradient descent on two-layer nets: Margin maximization and simplicity bias. *Adv. Neural Inf. Process. Syst.* 34 (2021), 12978–12991.
- Ma, C., Ying, L. On linear stability of SGD and input-smoothness of neural networks. *Adv. Neural Inf. Process. Syst.* 34 (2021), 16805–16817.
- Mulayoff, R., Michaeli, T. Unique properties of flat minima in deep networks. In *International Conference on Machine Learning* (2020), PMLR, 7108–7118.
- Mulayoff, R., Michaeli, T., Soudry, D. The implicit bias of minima stability: A view from function space. *Adv. Neural Inf. Process. Syst.* 34 (2021), 17749–17761.
- Nacson, M.S., Gunasekar, S., Lee, J., Srebro, N., Soudry, D. Lexicographic and depth-sensitive margins in homogeneous and non-homogeneous deep models. In *International Conference on Machine Learning* (2019), PMLR, 4683–4692.
- Nacson, M.S., Ravichandran, K., Srebro, N., Soudry, D. Implicit Bias of the Step Size in Linear Diagonal Neural Networks. In *International Conference on Machine Learning* (2022), PMLR, 16270–16295.
- Neyshabur, B., Bhojanapalli, S., McAllester, D., Srebro, N. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems* (2017), Curran Associates, Inc, Red Hook, NY, 5947–5958.
- Razin, N., Cohen, N. Implicit regularization in deep learning may not be explainable by norms. *Adv. Neural Inf. Process. Syst.* 33 (2020), 21174–21187.
- Safra, I., Vardi, G., Lee, J.D. On the effective number of linear regions in shallow univariate ReLU networks: Convergence guarantees and implicit bias. arXiv preprint arXiv:2205.09072 (2022).
- Sarussi, R., Brutzkus, A., Globerson, A. Towards understanding learning in neural networks with linear teachers. In *International Conference on Machine Learning* (2021), PMLR, 9313–9322.
- Shalev-Shwartz, S., Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.
- Soudry, D., Hoffer, E., Nacson, M.S., Gunasekar, S., Srebro, N. The implicit bias of gradient descent on separable data. *J. Mach. Learn. Res.* 19, 1 (2018), 2822–2878.
- Vardi, G., Shamir, O. Implicit regularization in relu networks with the square loss. In *Conference on Learning Theory* (2021), PMLR, 4224–4258.
- Vardi, G., Shamir, O., Srebro, N. On margin maximization in linear and ReLU networks. arXiv preprint arXiv:2110.02732 (2021).
- Woodworth, B., Gunasekar, S., Lee, J.D., Moroshko, E., Savarese, P., Golan, I., et al. Kernel and rich regimes in overparameterized models. In *Conference on Learning Theory* (2020), PMLR, 3635–3673.
- Wu, L., Ma, C., et al. How sgd selects the global minima in overparameterized learning: A dynamical stability perspective. *Adv. Neural Inf. Process. Syst.* 31 (2018).
- Wu, L., Wang, M., Su, W. When does SGD favor flat minima? A quantitative characterization via linear stability. arXiv preprint arXiv:2207.02628 (2022).
- Yun, C., Krishnan, S., Mobahi, H. A unifying view on implicit bias in training linear neural networks. In *International Conference on Learning Representations (ICLR)* (2021), OpenReview.net.
- Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM* 64, 3 (2021), 107–115.

Gal Vardi is a postdoctoral scholar at TTI-Chicago and the Hebrew University, Jerusalem, Israel.

Copyright held by author/owner.
Publication rights licensed to ACM.

ACM Distinguished Speakers

Talks by and with technology leaders and innovators

**A great speaker can make the difference
between a good event and a WOW event!**

ACM provides ACM Chapters, colleges and universities, corporations, event and conference planners, and agencies with direct access to top technology leaders and innovators from every sector of the computing industry.

The ACM Distinguished Speakers Program (DSP) features renowned thought leaders in industry, academia, and government speaking about the most important topics in computing and IT today.

ACM Local Chapters

Boost attendance at your meetings with live talks by DSP speakers and keep your chapter members informed of the latest industry findings.

Corporations

Educate your technical staff, ramp up the knowledge of your team, and give your employees the opportunity to have their questions answered by experts in their field.

Colleges and Universities

Expand the knowledge base of your students with exciting lectures and the chance to engage with a computing professional in their desired field of expertise.

Event and Conference Planners

Use the ACM DSP to help find compelling speakers for your next conference.

Book the speaker for your next virtual or in-person event through the ACM DSP program and deliver compelling and insightful content to your audience. ACM will cover the cost of transportation for the speaker to travel to your in-person event.

speakers.acm.org



Association for
Computing Machinery

Advancing Computing as a Science & Profession

If you have questions, please send them to **acmdsp@acm.org**.

P. 96

**Technical
Perspective
FoundationDB
Performs
Balancing Act**

By Alfons Kemper

P. 97

FoundationDB: A Distributed Key-Value Store

By Jingyu Zhou, Meng Xu, Alexander Shraer, Bala Namasivayam, Alex Miller, Evan Tschannen, Steve Atherton, Andrew J. Beamon, Rusty Sears, John Leach, Dave Rosenthal, Xin Dong, Will Wilson, Ben Collins, David Scherer, Alec Grieser, Yang Liu, Alvin Moore, Bhaskar Muppana, Xiaoge Su, and Vishesh Yadav

P. 106

**Technical
Perspective
What Does
Provable Security
Mean for
Cryptographic
Schemes?**

By David Pointcheval

P. 107

On the (In)Security of ElGamal in OpenPGP

By Luca De Feo, Bertram Poettering, and Alessandro Sorniotti

Technical Perspective

FoundationDB Performs Balancing Act

By Alfons Kemper

WE HAVE ARRIVED in the so-called Big Data era with data volumes growing exponentially. This requires effective data management systems with extreme scalability in size as well as speed, often coined “Internet scalability.” David Patterson (*Communications*, Oct. 2004) referred to an old network saying: “Bandwidth problems can be cured with money. Latency problems are harder because the speed of light is fixed—you can’t bribe God.” Therefore, reducing latency in a global application scenario requires distributing (that is, storing and/or replicating) data worldwide. Unfortunately, building efficient and reliable distributed data management systems remains an inherently difficult task.

With the emergence of (geographically) distributed data management in cloud infrastructures, the key value (KV) systems were promoted as NoSQL systems. To achieve maximum availability and performance, these KV stores sacrificed the “holy grail” of database consistency and relied on relaxed consistency models, such as eventual consistency. This was a consequence of Eric Brewer’s so-called CAP observation (aka Theorem)¹ stating that only two of the three desiderata of distributed systems could possibly be satisfied at the same time: Consistency, where every read operation receives the most recent committed write; availability which nowadays typically strives for the so-called many nines up-time (meaning, say, 99.99999% up-time); and, partition tolerance, which demands the system must remain operational even under severe network malfunctions.

Because of this conjecture, NoSQL system designers traded consistency for higher availability and network fault tolerance. The relaxed replica convergence models were categorized by Amazon’s Werner Vogels² in 2009. However, many mission-critical applications even in cloud settings do require stronger consistency guarantees. Eventual

consistency only ensures convergence to the same state of all replicas.

FoundationDB, as explored in the following paper, pioneered the development of a scalable distributed KV store with strong consistency guarantees. It started 10 years ago as an open source project and is now widely used as a mission-critical backbone repository in cloud infrastructures, such as Apple and Snowflake. In this respect, FoundationDB reunites the NoSQL paradigm of high availability and low latency with ACID (atomicity, consistency, isolation, durability) guarantees imposed by traditional database systems. This new breed of systems is therefore coined NewSQL—albeit not all offering SQL interfaces.

For scalability and elasticity in cloud infrastructures, FoundationDB exhibits a fully disaggregated architecture consisting of a storage system, a logging system, and a separated transaction system. Storage servers are decoupled from log servers, which maintain the “ground truth.” Storage servers continuously apply log records from log servers to lag only slightly behind the transaction commit state—in production system measurements the authors report a lag in the order of just milliseconds only. The transaction system employs an optimistic multi-version concurrency control scheme with a subsequent verification phase to ensure strict serializability. Thereby, read-only transactions are not penalized as they access versions that were committed by the time they started. For this purpose, a so-called sequencer process determines the corresponding time stamp that is then observed by proxies to access the correct version from storage servers.

For write transactions, the sequencer assigns the commit time stamp with which the resolvers verify strict serializability by comparing the transaction’s read set (that is, the key ranges of key value pairs that were accessed during

the transaction) with the write sets of transactions committed in the meantime. Successfully verified transactions are made durable by writing their logs to multiple log servers for fault tolerance. These log servers can be replicated across distinct (geographic) regions to tolerate failures (such as power outages) within an entire region. The FoundationDB system can be configured for synchronous as well as asynchronous log transferal—thereby trading off between safety and latency requirements. In case of a primary server failure the synchronous mode guarantees seamless instantaneous takeover by the secondary server—albeit incurring extra latency overhead during commit processing. Under asynchronous mode the primary server maintains several satellites within the same region which, in case of failure, can submit the log’s suffix that was not yet sent to the secondary server in a remote region. Thus, the trade-off between performance/responsiveness and failure resilience is one of the foremost goals of the FoundationDB design.

Correctness and robustness were also a key aspect in the development life cycle of the FoundationDB system. To achieve this, the team relied on simulation testing that allowed the distributed software code to run in a deterministic manner for error reproducibility and extensive test coverage.

To conclude, the design of FoundationDB lays the “foundation” for future cloud information systems that must balance performance and consistency requirements. 

References

1. Fox, A. and Brewer, E. Harvest, yield and scalable tolerant systems. In *Proceedings of the 7th Workshop Hot Topics in Operating Systems*. IEEE CS, 1999, 174–178.
2. Vogels, W. Eventually consistent. *Commun. ACM* 52 (2009), 40–44.

Alfons Kemper is a professor of database management systems and Head of the Computer Science Department of the Technical University of Munich, Germany.

Copyright held by author.

FoundationDB: A Distributed Key-Value Store

By Jingyu Zhou, Meng Xu, Alexander Shraer, Bala Namasivayam, Alex Miller, Evan Tschannen, Steve Atherton, Andrew J. Beamon, Rusty Sears, John Leach, Dave Rosenthal, Xin Dong, Will Wilson, Ben Collins, David Scherer, Alec Grieser, Yang Liu, Alvin Moore, Bhaskar Muppana, Xiaoge Su, and Vishesh Yadav

Abstract

FoundationDB is an open-source transactional key-value store created more than 10 years ago. It is one of the first systems to combine the flexibility and scalability of NoSQL architectures with the power of ACID transactions. FoundationDB adopts an unbundled architecture that decouples an in-memory transaction management system, a distributed storage system, and a built-in distributed configuration system. Each sub-system can be independently provisioned and configured to achieve scalability, high availability, and fault tolerance. FoundationDB includes a deterministic simulation framework, used to test every new feature under a myriad of possible faults. This rigorous testing makes FoundationDB extremely stable and allows developers to introduce and release new features in a rapid cadence. FoundationDB offers a minimal and carefully chosen feature set, which has enabled a range of disparate systems to be built as layers on top. FoundationDB is the underpinning of cloud infrastructure at Apple, Snowflake, and other companies, due to its consistency, robustness, and availability for storing user data, system metadata and configuration, and other critical information.

1. INTRODUCTION

Many cloud services rely on scalable, distributed storage backends for persisting application state. Such storage systems must be fault tolerant and highly available, and at the same time provide sufficiently strong semantics and flexible data models to enable rapid application development. Such services must scale to billions of users, petabytes or exabytes of stored data, and millions of requests per second.

More than a decade ago, NoSQL storage systems emerged offering ease of application development, making it simple to scale and operate storage systems, offering fault-tolerance and supporting a wide range of data models (instead of the traditional rigid relational model). In order to scale, these systems sacrificed transactional semantics, and instead provided eventual consistency, forcing application developers to reason about interleavings of updates from concurrent operations.

FoundationDB (FDB)³ was created in 2009 and gets its name from the focus on providing what we saw as the foundational set of building blocks required to build higher-level distributed systems. It is an ordered, transactional, key-value store natively supporting multi-key strictly serializable transactions across its entire key space. Unlike most databases, which bundle together a storage engine, data model, and query language, forcing users to choose

all three or none, FDB takes a modular approach: it provides a highly scalable, transactional storage engine with a minimal yet carefully chosen set of features. It provides no structured semantics, no query language, data model or schema management, secondary indices, or many other features one normally finds in a transactional database. Offering these would benefit some applications but others that do not require them (or do so in a slightly different form) would need to work around them. Instead, The NoSQL model leaves application developers with great flexibility. Applications can manage data stored as simple key-value pairs, but at the same time implement advanced features, such as consistent secondary indices and referential integrity checks.¹⁰ FDB defaults to strictly serializable transactions but allows relaxing these semantics for applications that don't require them with flexible, fine-grained controls over conflicts.

One of the reasons for its popularity and growing open source community is FoundationDB's focus on the "lower half" of a database, leaving the rest to its "layers"—stateless applications developed on top to provide various data models and other capabilities. With this, applications that would traditionally require completely different types of storage systems, can instead all leverage FDB. Indeed, the wide range of layers that have been built on FDB in recent years is evidence of the usefulness of this unusual design. For example, the FoundationDB Record Layer¹⁰ adds back much of what users expect from a relational database, and JanusGraph,⁶ a graph database, provides an implementation as a FoundationDB layer.⁵ In its newest release, CouchDB¹ (arguably the first NoSQL system) is being rebuilt as a layer on top of FoundationDB.

Testing and debugging distributed systems is at least as hard as building them. Unexpected process and network failures, message reorderings, and other sources of non-determinism can expose subtle bugs and implicit assumptions that break in reality, which are extremely difficult to reproduce or debug. The consequences of such subtle bugs are especially severe for database systems, which purport to offer perfect fidelity to an unambiguous contract. Moreover, the stateful nature of a database system means that any such bug can result in subtle data corruption that

The original version of this paper, entitled "FoundationDB: A Distributed Unbundled Transactional Key Value Store," was published in SIGMOD '21, June 20–25, 2021, Virtual Event, China; <https://doi.org/10.1145/3448016.3457559>

may not be discovered for months. Model-checking techniques can verify the correctness of distributed protocols but often fall short of checking the actual implementation. Deep bugs,¹⁸ which only happen when multiple crashes or restarts occur in a particular sequence, pose a challenge even for end-to-end testing infrastructure. FDB took a radical approach—before building the database itself, we built a deterministic database simulation framework that can simulate a network of interacting processes and a variety of disk, process, network, and request-level failures and recoveries, all within a single physical process. A syntactic extension to C++, called Flow,² was created specifically for this purpose. This rigorous testing in simulation makes FDB extremely stable and allows its developers to introduce new features and releases in a rapid cadence.

FDB adopts an unbundled architecture¹⁹ that comprises a control plane and a data plane. The control plane manages the metadata of the cluster and uses Active Disk Paxos⁹ for high availability. The data plane consists of a transaction management system, responsible for processing updates, and a distributed storage layer serving reads; both can be independently scaled out. FDB achieves strict serializability through a combination of optimistic concurrency control (OCC)¹⁷ and multi-version concurrency control (MVCC).⁸ One of the features distinguishing FDB from other distributed databases is its approach to handling failures. Unlike most similar systems, FDB does not rely on quorums to mask failures, but rather tries to eagerly detect and recover from them by reconfiguring the system. This allows us to achieve the same level of fault tolerance with significantly fewer resources: FDB can tolerate f failures with only $f + 1$ (rather than $2f + 1$) replicas. This approach is best suited for deployments in a local or metro area. For WAN deployments, FDB offers a novel strategy that avoids cross-region write latencies while providing automatic failover between regions without losing data.

This paper makes three primary contributions. First, we describe an open-source distributed storage system, FoundationDB, combining NoSQL and ACID, used in production at Apple, Snowflake, and other companies. Second, an integrated deterministic simulation framework makes FoundationDB one of the most stable systems of its kind. Third, we describe a unique architecture and approach to transaction processing, fault tolerance, and high availability.

2. DESIGN

The main design principles of FDB are:

- *Divide-and-Conquer (or separation of concerns).* FDB decouples the transaction management system (write path) from the distributed storage (read path) and scales them independently. Within the transaction management system, processes are assigned various roles representing different aspects of transaction management. Furthermore, cluster-wide orchestrating tasks, such as overload control and load balancing are also divided and serviced by additional heterogeneous roles.

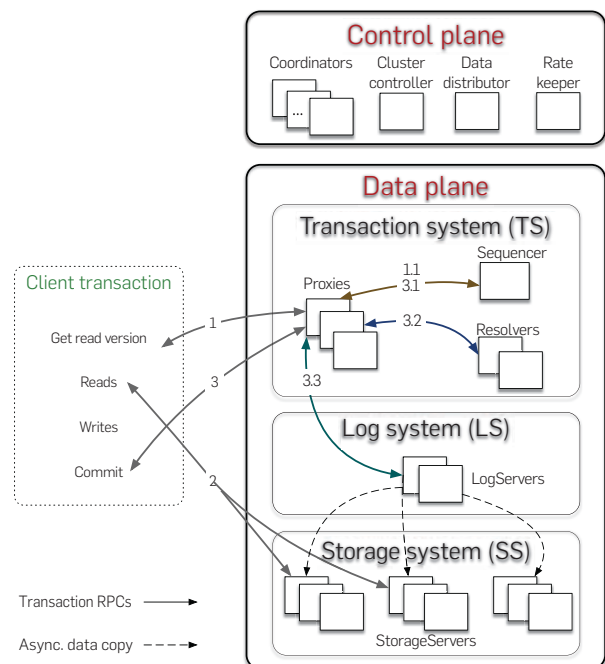
- *Make failure a common case.* For distributed systems, failure is a norm rather than an exception. To cope with failures in the transaction management system of FDB, we handle all failures through the recovery path: the transaction system proactively shuts down when it detects a failure. Thus, all failure handling is reduced to a single recovery operation, which becomes a common and well-tested code path. To improve availability, FDB strives to minimize Mean-Time-To-Recovery (MTTR). In our production clusters, the total time is usually less than five seconds.
- *Simulation testing.* FDB relies on a randomized, deterministic simulation framework for testing the correctness of its distributed database. Simulation tests not only expose deep bugs,¹⁸ but also boost developer productivity and the code quality of FDB.

2.1. Architecture

An FDB cluster has a control plane for managing critical system metadata and cluster-wide orchestration, and a data plane for transaction processing and data storage, as illustrated in Figure 1.

Control plane. The control plane is responsible for persisting critical system metadata, that is, the configuration of transaction systems, on Coordinators. These Coordinators form a Paxos group⁹ and elect a ClusterController. The ClusterController monitors all servers in the cluster and recruits three processes, Sequencer (described in Section 2.1.2), DataDistributor, and Ratekeeper, which are re-recruited if they fail or crash. The DataDistributor is responsible for monitoring failures and balancing data

Figure 1. Architecture and transaction processing.



among StorageServers. Ratekeeper provides overload protection for the cluster.

Data plane. FDB targets OLTP workloads that are read-mostly, read and write a small set of keys per transaction, have low contention, and require scalability. FDB chooses an unbundled architecture¹⁹: a distributed transaction management system (TS) consists of a Sequencer, Proxies, and Resolvers, all of which are stateless processes. A log system (LS) stores Write-Ahead-Log (WAL) for TS, and a separate distributed storage system (SS) is used for storing data and servicing reads. The LS contains a set of LogServers and the SS has a number of StorageServers. This scales well with Apple's largest transactional workloads.¹⁰

The Sequencer assigns a read and a commit version to each transaction. Proxies offer MVCC read versions to clients and orchestrate transaction commits. Resolvers check for conflicts among transactions. LogServers act as replicated, sharded, distributed persistent queues, each queue storing WAL data for a StorageServer.

The SS consists of a number of StorageServers, each storing a set of data shards, that is, contiguous key ranges, and serving client reads. StorageServers are the majority of processes in the system, and together they form a distributed B-tree. Currently, the storage engine on each StorageServer is an enhanced version of SQLite,¹⁵ with enhancements that make range clears faster, defer deletion to a background task, and add support for asynchronous programming.

Read-write separation and scaling. As mentioned above, processes are assigned different roles; FDB scales by adding new processes for each role. Clients read from sharded StorageServers, so reads scale linearly with the number of StorageServers. Writes are scaled by adding more Proxies, Resolvers, and LogServers. The control plane's singleton processes (e.g., ClusterController and Sequencer) and Coordinators are not performance bottlenecks; they only perform limited metadata operations.

Bootstrapping. FDB has no dependency on external coordination services. All user data and most system metadata (keys that start with 0xFF prefix) are stored in StorageServers. The metadata about StorageServers is persisted in LogServers, and the LogServers configuration data is stored in all Coordinators. The Coordinators are a disk Paxos group; servers attempt to become the ClusterController if one does not exist. A newly elected ClusterController reads the old LS configuration from the Coordinators and spawns a new TS and LS. Proxies recover system metadata from the old LS, including information about all StorageServers. The Sequencer waits until the new TS finishes recovery (Section 2.2.4), then writes the new LS configuration to all Coordinators. The new transaction system is then ready to accept client transactions.

Reconfiguration. The Sequencer process monitors the health of Proxies, Resolvers, and LogServers. Whenever there is a failure in the TS or LS, or the database configuration changes, the Sequencer terminates. The ClusterController detects the Sequencer failure, then recruits and bootstraps a new TS and LS. In this way, transaction processing is divided into epochs, where each epoch represents a generation of the transaction management system with its own Sequencer.

2.2. Transaction management

This section describes end-to-end transaction processing and strict serializability, then discusses logging and recovery.

End-to-end transaction processing. As illustrated in Figure 1, a client transaction starts by contacting one of the Proxies to obtain a read version (i.e., a timestamp). The Proxy then asks the Sequencer for a read version that is at least as large as all previously issued transaction commit versions, and sends this read version back to the client. The client may then issue reads to StorageServers and obtain values at that specific read version. Client writes are buffered locally without contacting the cluster and read-your-write semantics are preserved by combining results from database look-ups with uncommitted writes of the transaction. At commit time, the client sends the transaction data, including the read and write sets (i.e., key ranges), to one of the Proxies and waits for a commit or abort response. If the transaction cannot commit, the client may choose to restart it.

A Proxy commits a client transaction in three steps. First, it contacts the Sequencer to obtain a commit version that is larger than any existing read versions or commit versions. The Sequencer chooses the commit version by advancing it at a rate of one million versions per second. Then, the Proxy sends the transaction information to range-partitioned Resolvers, which implement FDB's optimistic concurrency control by checking for *read-write* conflicts. If all Resolvers return with no conflict, the transaction can proceed to the final commit stage. Otherwise, the Proxy marks the transaction as aborted. Finally, committed transactions are sent to a set of LogServers for persistence. A transaction is considered committed after all designated LogServers have replied to the Proxy, which reports the committed version to the Sequencer (to ensure that later transactions' read versions are after this commit) and then replies to the client. StorageServers continuously pull mutation logs from LogServers and apply committed updates to disks.

In addition to the above *read-write transactions*, FDB also supports *read-only transactions* and *snapshot reads*. A read-only transaction in FDB is both serializable (happens at the read version) and performant (thanks to the MVCC), and the client can commit these transactions locally without contacting the database. This is particularly important because the majority of transactions are read-only. Snapshot reads in FDB selectively relax the isolation

Before 7.1 release, the ClusterController delegates this work to the new Sequencer.

property of a transaction by reducing conflicts, that is, concurrent writes will not conflict with snapshot reads.

Strict serializability. FDB implements Serializable Snapshot Isolation (SSI) by combining OCC with MVCC. Recall that a transaction T_x gets both its read version and commit version from the Sequencer, where the read version is guaranteed to be no less than any committed version when T_x starts and the commit version is larger than any existing read or commit versions. This commit version defines a serial history for transactions and serves as a Log Sequence Number (LSN). Because T_x observes the results of all previously committed transactions, FDB achieves strict serializability. To ensure there are no gaps between LSNs, the Sequencer returns the previous commit version (i.e., previous LSN) with each commit version. A Proxy sends both LSN and the previous LSN to Resolvers and LogServers so that they can serially process transactions in the order of LSNs. Similarly, StorageServers pull log data from LogServers in increasing LSN order.

Resolvers use a lock-free conflict detection algorithm similar to *write-snapshot isolation*,²² with the difference that in FDB the commit version is chosen before conflict detection. This allows FDB to efficiently batch-process both version assignments and conflict detection.

The entire key space is divided among Resolvers allowing conflict detection to be performed in parallel. A transaction can commit only when all Resolvers admit the transaction. Otherwise, the transaction is aborted. It is possible that an aborted transaction is admitted by a subset of Resolvers, and they have already updated their history of potentially committed transactions, which may cause other transactions to conflict (i.e., a false positive). In practice, this has not been an issue for our production workloads, because transactions' key ranges usually fall into one Resolver. Additionally, because the modified keys expire after the MVCC window, such false positives are limited to only happen within the short MVCC window time (i.e., 5 seconds).

The OCC design of FDB avoids the complicated logic of acquiring and releasing (logical) locks, which greatly simplifies interactions between the TS and the SS. The price is wasted work done by aborted transactions. In our multitenant production workload transaction conflict rate is very low (less than 1%) and OCC works well. If a conflict happens, the client can simply restart the transaction.

Logging protocol. After a Proxy decides to commit a transaction, it sends a message to all LogServers: mutations are sent to LogServers responsible for the modified key ranges, while other LogServers receive an empty message body. The log message header includes both the current and previous LSN obtained from the Sequencer, as well as the largest known committed version (KCV) of this Proxy. LogServers reply to the Proxy once the log data is made durable, and the Proxy updates its KCV to the LSN if all replica LogServers have replied and this LSN is larger than the current KCV.

Shipping the redo log from the LS to the SS is not a part of the commit path and is performed in the background.

In FDB, StorageServers apply non-durable redo logs from LogServers to an in-memory index. In the common case, this happens before any read versions that reflect the commit is handed out to a client, allowing very low latency for serving multi-version reads. Therefore, when client read requests reach StorageServers, the requested version (i.e., the latest committed data) is usually already available. If fresh data is not available to read at a StorageServer replica, the client either waits for the data to become available or reissues the request at another replica.¹² If both read time out, the client can simply restart the transaction.

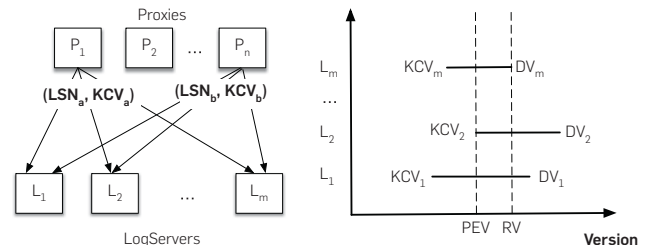
Since log data is already durable on LogServers, StorageServers can buffer updates in memory and persist batches of data to disks periodically, thus improving I/O efficiency.

Transaction system recovery. Traditional database systems often employ the ARIES recovery protocol.²⁰ During recovery, the system processes redo log records from the last checkpoint by re-applying them to relevant data pages. This brings the database to a consistent state; transactions that were in flight during the crash can be rolled back by executing the undo log records.

In FDB, recovery is purposely made very cheap—there is no need to apply undo log entries. This is possible because of a greatly simplifying design choice: redo log processing is the same as the normal log forward path. In FDB, StorageServers pull logs from LogServers and apply them in the background. The recovery process starts by detecting a failure and recruiting a new transaction system. The new TS can accept transactions before all the data in the old LogServers is processed. Recovery only needs to find out the end of the redo log: At that point (as in normal forward operation) StorageServers asynchronously replay the log.

For each epoch, the ClusterController executes recovery in several steps. First, it reads the previous TS configuration from Coordinators and locks this information to prevent another concurrent recovery. Next, it recovers previous TS system states, including information about older LogServers, stops them from accepting transactions, and recruits a new set of Sequencer, Proxies, Resolvers, and LogServers. After previous LogServers are stopped and a new TS is recruited, the ClusterController writes the new TS information to the Coordinators. Because Proxies and Resolvers are stateless, their

Figure 2. An illustration of RV and PEV.



recoveries have no extra work. In contrast, LogServers save the logs of committed transactions, and we need to ensure all such transactions are durable and retrievable by StorageServers.

The essence of the recovery of old LogServers is to determine the end of the redo log, that is, a Recovery Version (RV). Rolling back undo the log is essentially discarding any data after RV in the old LogServers and StorageServers. Figure 2 illustrates how RV is determined by the Sequencer. Recall that a Proxy request to LogServers piggybacks its KCV, the maximum LSN that this Proxy has committed, along with the LSN of the current transaction. Each LogServer keeps the maximum KCV received and a Durable Version (DV), which is the maximum LSN persisted by the LogServer (DV is ahead of KCV since it corresponds to in-flight transactions). During recovery, the Sequencer attempts to stop all m old LogServers, where each response contains the DV and KCV on that LogServer. Assume the replication degree for LogServers is k . Once the Sequencer has received more than $m - k$ replies, the Sequencer knows the previous epoch has committed transactions up to the maximum of all KCVs, which becomes the previous epoch's end version (PEV). All data before this version has been fully replicated. For the current epoch, its start version is $PEV + 1$ and the Sequencer chooses the minimum of all DVs to be the RV. Logs in the range of $[PEV + 1, RV]$ are copied from the previous epoch's LogServers to the current ones, for healing the replication degree in case of LogServer failures. The overhead of copying this range is very small because it only contains a few seconds' log data.

When Sequencer accepts new transactions, the first is a special recovery transaction that informs StorageServers of the RV so that they can roll back any data larger than the RV. The current FDB storage engine consists of an unversioned SQLite¹⁵ B-tree and in-memory multi-versioned redo log data. Only mutations leaving the MVCC window (i.e., committed data) are written to SQLite. The rollback is simply discarding in-memory multi-versioned data in StorageServers. Then StorageServers pull any data larger than version PEV from new LogServers.

2.3. Replication

FDB uses a combination of various replication strategies for different data to tolerate f failures:

- *Metadata replication.* System metadata of the control plane is stored on Coordinators using Active Disk Paxos.⁹ As long as a quorum (i.e., majority) of Coordinators are live, this metadata can be recovered.
- *Log replication.* When a Proxy writes logs to LogServers, each sharded log record is synchronously replicated on $k = f + 1$ LogServers. Only when all k have replied with successful persistence can the Proxy send back the commit response to the client. Failure of LogServer results in a transaction system recovery.
- *Storage replication.* Every shard, that is, a key range, is asynchronously replicated to $k = f + 1$ StorageServers, which is called a *team*. A StorageServer usually hosts

a number of shards so that its data is evenly distributed across many teams. A failure of a StorageServer triggers DataDistributor to move data from teams containing the failed process to other healthy teams.

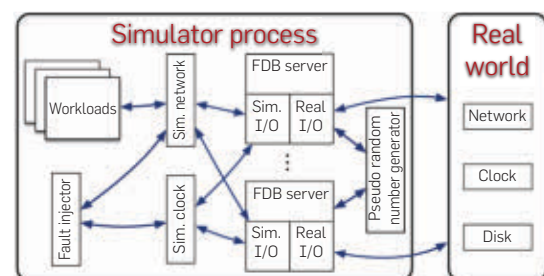
Note the storage team abstraction is more sophisticated than Copysets.¹¹ To reduce the chance of data loss due to simultaneous failures, FDB ensures that at most one process in a replica group is placed in a *fault domain*, for example, a host, rack, or availability zone. Each team is guaranteed to have at least one process live and there is no data loss if any one of the respective fault domains remains available.

3. SIMULATION TESTING

Testing and debugging distributed systems is a challenging and inefficient process. This problem is particularly acute for FDB—any failure of its strong concurrency control contract can produce almost arbitrary corruption in systems layered on top. Accordingly, an ambitious approach to end-to-end testing was adopted from the beginning: the real database software is run, together with randomized synthetic workloads and fault injection, in a deterministic discrete-event simulation. The harsh simulated environment quickly provokes bugs in the database, and determinism guarantees that every such bug can be reproduced and investigated.

Deterministic simulator. FDB was built from the ground up to enable this testing approach. All database code is deterministic and multithreaded concurrency is avoided (instead, one database node is deployed per core). Figure 3 illustrates the simulator process of FDB, where all sources of nondeterminism and communication are abstracted, including network, disk, time, and pseudo-random number generator. FDB is written in Flow,² a novel syntactic extension to C++ adding async/await-like concurrency primitives with automatic cancellation, permitting highly concurrent code to execute deterministically. Flow provides the Actor programming model⁷ that abstracts various actions of the FDB server process into a number of actors that are scheduled by the Flow runtime library. The simulator process is able to spawn multiple FDB servers that communicate with each other through a simulated network in a single discrete-event simulation. The production implementation is a simple shim to the relevant system calls.

Figure 3. The FDB deterministic simulator.



The simulator runs multiple workloads (written in Flow) that communicate with simulated FDB servers through the simulated network. These workloads include fault injection instructions, mock applications, database configuration changes, and internal database functionality invocations. Workloads are composable to exercise various features and are reused to construct comprehensive test cases.

Test oracles. FDB uses a variety of test oracles to detect failures in simulation. Most of the synthetic workloads have assertions built in to verify the contracts and properties of the database, for example, by checking invariants in their data that can only be maintained through transaction atomicity and isolation. Assertions are used throughout the code base to check properties that can be verified “locally.” Properties like recoverability (eventual availability) can be checked by returning the modeled hardware environment (after a set of failures sufficient to break the database’s availability) to a state in which recovery should be possible and verifying that the cluster eventually recovers.

Fault injection. Simulation injects machine, rack, and data-center failures and reboots, a variety of network faults, partitions, and latency problems, disk behavior (e.g., the corruption of unsynchronized writes when machines reboot), and randomizes event times. This variety of fault injection both tests the database’s resilience to specific faults and increases the diversity of states in simulation. Fault injection distributions are carefully tuned to avoid driving the system into a small state space caused by an excessive fault rate.

FDB itself cooperates with the simulation in making rare states and events more common, through a high-level fault injection technique informally referred to as “buggification.” At many places in its code base, the simulation is allowed to inject some unusual (but not contract-breaking) behavior such as unnecessarily returning an error from an operation that usually succeeds, injecting a delay in an operation that is usually fast, or choosing an unusual value for a tuning parameter, etcetera. This complements fault injection at the network and hardware levels. Randomization of tuning parameters also ensures that specific performance tuning values do not accidentally become necessary for correctness.

Swarm testing¹⁴ is extensively used to maximize the diversity of simulation runs. Each run uses a random cluster size and configuration, random workloads, random fault injection parameters, random tuning parameters, and enables and disables a random subset of buggification points. We have open-sourced the swarm testing framework for FDB.⁴

Conditional coverage macros are used to evaluate and tune the effectiveness of the simulation. For example, a developer concerned that a new piece of code may rarely be invoked with a full buffer can add the line `TEST (buffer.is_full());` and analysis of simulation results will tell them how many distinct simulation runs achieved that condition. If the number is too low, or zero, they can add buggification, workload, or fault injection functionality to ensure that scenario is adequately tested.

Latency to bug discovery. Finding bugs quickly is important both so that they are encountered in testing before production, and for engineering productivity (since bugs found immediately in an individual commit can be trivially traced to that commit). Discrete-event simulation can run arbitrarily faster than real-time if CPU utilization within the simulation is low, as the simulator can fast-forward clock to the next event. Many distributed systems bugs take time to play out, and running simulations with long stretches of low utilization allows many more of these to be found per core second than in “real-world” end-to-end tests.

Additionally, randomized testing is embarrassingly parallel and FDB developers can and do “burst” the amount of testing they do before major releases, in the hopes of catching exceptionally rare bugs that have thus far eluded the testing process. Since the search space is effectively infinite, running more tests results in more code being covered and more potential bugs being found, in contrast to scripted functional or system testing.

Limitations. Simulation cannot reliably detect performance issues, such as an imperfect load-balancing algorithm. It is also unable to test third-party libraries or dependencies, or even first-party code not implemented in Flow. As a consequence, we have largely avoided taking dependencies on external systems. Finally, bugs in critical dependent systems, such as a filesystem or the operating system, or misunderstandings of their contract, can lead to bugs in FDB. For example, several bugs have resulted from the true operating system contract being weaker than it was believed to be.

4. EVALUATION

This section studies the scalability of FDB and provides some data on the time of reconfiguration.

4.1. Scalability test

The experiments were conducted on a test cluster of 27 machines in a single data center. Each machine has a 16-core 2.5 GHz Intel Xeon CPU with hyper-threading enabled, 256 GB memory, 8 SSD disks, connected via 10 Gigabit Ethernet. Each machine runs 14 StorageServers on 7 SSD disks and reserves the remaining SSD for LogServer. In the experiments, we use the same number of Proxies and LogServers. The replication degrees for both LogServers and StorageServers are set to three.

We use a synthetic workload to evaluate the performance of FDB. Specifically, there are four types of transactions: (1) *blind writes* that update a configured number of random keys; (2) *range reads* that fetch a configured number of continuous keys starting at a random key; (3) *point reads* that fetch 10 random keys; and (4) *point writes* that fetch 5 random keys and update another 5 random keys. We use blind writes and range reads to evaluate the write and read performance, respectively. Point reads and point writes are used together to evaluate mixed read-write performance. For instance, 90% reads and 10% writes (90/10 read-write) is constructed with 80% point reads and 20%

point writes transactions. Each key is 16 bytes and the value size is uniformly distributed between 8 and 100 bytes (averaging 54 bytes). The database is pre-populated with data using the same size distribution. In the experiments, we make sure the dataset cannot be completely cached in the memory of StorageServers.

Figure 4 illustrates the scalability test of FDB from 4 to 24 machines using 2 to 22 Proxies or LogServers. Figure 4a shows that the write throughput scales from 67 to 391 MBps (5.84X) for 100 operations per transaction (T100), and from 73 to 467 MBps (6.40X) for 500 operations per transaction (T500). Note the raw write throughput is three times higher because each write is replicated three times to LogServers and StorageServers. LogServers are CPU saturated at the maximum write throughput. Read throughput increases from 2946 to 10,096 MBps (3.43X) for T100, and from 5055 to 21,830 MBps (4.32X) for T500, where StorageServers are saturated. For both reads and writes, increasing the number of operations in a transaction boosts throughput. However, increasing operations further (e.g., to 1000) doesn't bring significant changes. Figure 4b shows the operations per second for 90/10 read-write traffic, which increases from 593k to 2779k (4.69X). In this case, Resolvers and Proxies are CPU saturated.

The above experiments study saturated performance. Figure 5 illustrates the client performance on a 24-machine cluster with varying operation rates of 90/10 read-write load. This configuration has 2 Resolvers, 22 LogServers, 22 Proxies, and 336 StorageServers. Figure 5a shows that the throughput scales linearly with more operations per second (Ops) for both reads and writes. For latency, Figure 5b shows that when Ops is below 100k, the mean

latencies remain stable: about 0.35 ms to read a key, 2 ms to commit, and 1ms to get a read version (GRV). Read is a single-hop operation, thus is faster than the two-hop GRV request. The commit request involves multiple hops and persistence to three LogServers, thus higher latency than reads and GRVs. When Ops exceeds 100 k, all these latencies increase because of more queuing time. At 2m Ops, Resolvers and Proxies are saturated. Batching helps to sustain the throughput but commits latency spike to 368 ms due to saturation.

4.2. Reconfiguration duration

We collected 289 reconfigurations (i.e., transaction system recovery) traces from our production clusters that typically host hundreds of TBs data. Because of the client-facing nature, short reconfiguration time is critical for the high availability of these clusters. Figure 6 illustrates the cumulative distribution function (CDF) of the reconfiguration times. The median and 90-percentile are 3.08 and 5.28 seconds, respectively. The reason for these short recovery times is that they are not bounded by the data or transaction log size, and are only related to the system metadata sizes. During the recovery, read-write transactions were temporarily blocked and were retried after the timeout. However, client reads were not impacted. The causes of these reconfigurations include automatic failure recovery from software or hardware faults, software upgrades, database configuration changes, and the manual mitigation of production issues.

5. LESSONS LEARNED

This section discusses our experience and lessons of FDB.

5.1. Architecture design

The divide-and-conquer design principle has proven to be an enabling force for flexible cloud deployment, making the database extensible as well as performant. First, separating the transaction system from the storage layer enables greater flexibility in placing and scaling compute and storage resources independently. An added benefit of LogServers is that they are akin to witness replicas; in some of our multi-region production deployments, LogServers significantly reduce the number of StorageServers (full replicas) required to achieve the same high-availability properties. Further, operators are free to place heterogeneous roles of FDB on different server instance types, optimizing for performance and costs.

Figure 4. Scalability test.

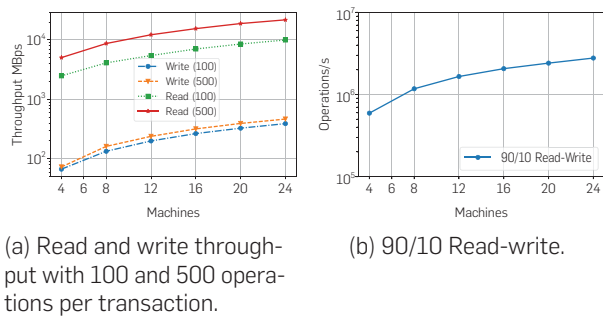


Figure 5. Throughput and average latency for different operation rates on a 24-machine cluster configuration.

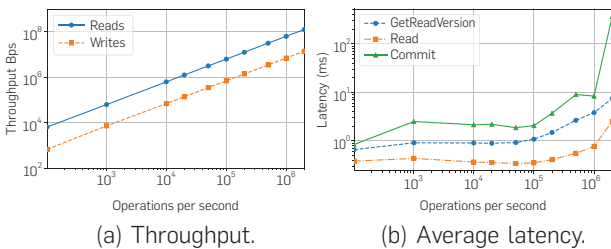
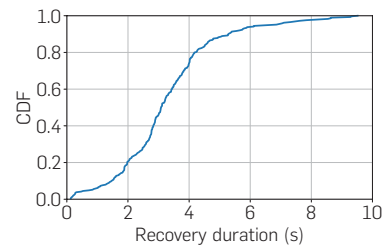


Figure 6. CDF plot for reconfiguration duration in seconds.



Second, the decoupling design makes it possible to extend the database functionality, such as our ongoing work of supporting RocksDB¹³ as a drop-in replacement for the current SQLite engine. Finally, many of the recent performance improvements are specializing functionality as dedicated roles, for example, separating DataDistributor and Ratekeeper from Sequencer, adding storage cache, dividing Proxies into get-read-version Proxy and commit Proxy. This design pattern successfully allows new features and capabilities to be added frequently.

5.2. Simulation testing

Simulation testing has enabled FDB to maintain a very high development velocity with a small team by shortening the latency between a bug being introduced and a bug being found, and by allowing deterministic reproduction of issues. Adding additional logging, for instance, generally does not affect the deterministic ordering of events, so an exact reproduction is guaranteed. The productivity of this debugging approach is so much higher than normal production debugging, that in the rare circumstances when a bug was first found “in the wild,” the debugging process was almost always first to improve the capabilities or the fidelity of the simulation until the issue could be reproduced there, and only then to begin the normal debugging process. Rigorous correctness testing via simulation makes FDB extremely reliable. In the past several years, CloudKit²¹ has deployed FDB for more than 0.5M disk years without a single data corruption event.

It is hard to measure the productivity improvements stemming from increased confidence in the testability of the system. On numerous occasions, the FDB team executed ambitious, ground-up rewrites of major subsystems. Without simulation testing, many of these projects would have been deemed too risky or too difficult, and not even attempted.

The success of simulation has led us to continuously push the boundary of what is amenable to simulation testing by eliminating dependencies and reimplementing them ourselves in Flow. For example, early versions of FDB depended on Apache Zookeeper for coordination, which was deleted after real-world fault injection found two independent bugs in Zookeeper (circa 2010) and was replaced by a de novo Paxos implementation written in Flow. No production bugs have ever been reported since.

5.3. Fast recovery

Fast recovery is not only useful for improving availability but also greatly simplifies software upgrades and configuration changes and makes them faster. The traditional wisdom of upgrading a distributed system is to perform rolling upgrades so that rollback is possible when something goes wrong. The duration of rolling upgrades can last from hours to days. In contrast, FoundationDB upgrades can be performed by restarting all processes at the same time, which usually finishes within a few seconds. Because this upgrade path has been extensively tested in simulation, all upgrades in Apple’s production clusters are performed in this way. Additionally, this upgrade path simplifies protocol compatibility between different versions—we only need to make sure on-disk data is compatible. There is no

need to ensure the compatibility of RPC protocols between different software versions.

An interesting discovery is that fast recovery sometimes can automatically heal latent bugs, which is similar to software rejuvenation.¹⁶ For instance, after we separated the DataDistributor role from the Sequencer, we were surprised to discover several unknown bugs in the DataDistributor. This is because, before the change, DataDistributor is restarted with Sequencer, which effectively reinitializes and heals the states of the DataDistributor. After the separation, we made DataDistributor a long-running process independent of transaction system recovery (including Sequencer restart). As a result, the erroneous states of the DataDistributor are never healed and cause test failures.

5.4. 5s MVCC window

FDB chooses a 5-second MVCC window to limit the memory usage of the transaction system and storage servers because the multi-version data is stored in the memory of Resolvers and StorageServers, which in turn restricts transaction sizes. From our experience, this 5s window is long enough for the majority of OLTP use cases. If a transaction exceeds the time limit, it is often the case that the client application is doing something inefficient, for example, issuing reads one by one instead of parallel reads. As a result, exceeding the time limit often exposes inefficiency in the application.

For some transactions that may span more than 5s, many can be divided into smaller transactions. For instance, the continuous backup process of FDB will scan through the key space and create snapshots of key ranges. Because of the 5s limit, the scanning process is divided into a number of smaller ranges so that each range can be performed within 5s. In fact, this is a common pattern: one transaction creates a number of jobs and each job can be further divided or executed in a transaction. FDB has implemented such a pattern in an abstraction called TaskBucket and the backup system heavily depends on it.

6. CONCLUSION

FoundationDB is a key value store designed for OLTP cloud services. The main idea is to decouple transaction processing from logging and storage. Such an unbundled architecture enables the separation and horizontal scaling of both read and write handling. The transaction system combines OCC and MVCC to ensure strict serializability. The decoupling of logging and the determinism in transaction orders greatly simplify recovery, thus, allowing unusually quick recovery time and improving availability. Finally, deterministic and randomized simulation has ensured the correctness of the database implementation.

Acknowledgments

FoundationDB is still an ongoing effort. We thank Zhe Wu, He Liu, Dan Lambright, Hao Fu, Neethu H. Bingi, Renxuan Wang, Sreenath Bodagala, Yao Xiao, Aaron Molitor, our SRE team, and the Snowflake team for their continuous improvement to the project.



References

1. CouchDB. <https://couchdb.apache.org/>.
2. Flow. <https://github.com/apple/foundationdb/tree/master/flow>.
3. FoundationDB. <https://github.com/apple/foundationdb>.
4. FoundationDB Joshua. <https://github.com/FoundationDB/fdb-joshua>.
5. Foundationdb storage adapter for janusgraph. <https://github.com/JanusGraph/janusgraph-foundationdb>.
6. Janusgraph. <https://janusgraph.org/>.
7. Agha, G. *Actors: A Model of Concurrent Computation in Distributed Systems*. MIT Press Cambridge, MA, USA, 1986.
8. Bernstein, P.A., Hadzilacos, V., Goodman, N. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley Boston, MA, USA, 1987.
9. Chockler, G., Malkhi, D. Active disk paxos with infinitely many processes. In *ACM PODC* (2002).
10. Chrysafis, C., Collins, B., Dugas, S., Dunkelberger, J., Ehsan, M., Gray, S., et al. FoundationDB record layer: A multi-tenant structured datastore. In *ACM SIGMOD* (2019).
11. Cidon, A., Rumble, S., Stutsman, R., Katti, S., Ousterhout, J., Rosenblum, M. Copysets: Reducing the frequency of data loss in cloud storage. In *USENIX Annual Technical Conference* (2013).
12. Dean, J., Barroso, L.A. The tail at scale. *Commun. ACM* 56, 2 (Feb. 2013), 74–80.
13. Facebook. Rocksdb. <https://rocksdb.org>.
14. Groce, A., Zhang, C., Eide, E., Chen, Y., Regehr, J. Swarm testing. In *ACM ISSTA* (2012).
15. Hipp, R.D. SQLite. 2020. <https://www.sqlite.org/index.html>.
16. Huang, Y., Kintala, C., Kolettis, N., Fulton, N.D. Software rejuvenation: Analysis, module and applications. In *Twenty-Fifth International Symposium on Fault-Tolerant Computing* (1995), 381–390.
17. Kung, H.T., Robinson, J.T. On optimistic methods for concurrency control. *ACM Trans. Database Syst.* 6, 2 (1981), 213–226.
18. Leesatapornwongsa, T., Hao, M., Joshi, P., Lukman, J.F., Gunawi, H.S. Samc: Semantic-aware model checking for fast discovery of deep bugs in cloud systems. In *USENIX OSDI* (2014).
19. Lomet, D., Fekete, A., Weikum, G., Zwilling, M.J. Unbundling transaction services in the cloud. In *CIDR* (2009).
20. Mohan, C., Haderle, D., Lindsay, B.G., Pirahesh, H., Schwarz, P.M. Aries: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. *ACM Trans. Database Syst.* 17, 1 (1992), 94–162.
21. Shraer, A., Aybes, A., Davis, B., Chrysafis, C., Browning, D., Krugler, E., et al. Cloudkit: Structured storage for mobile applications. *Proc. VLDB Endow.* 11, 5 (Jan. 2018) 540–552.
22. Yabandeh, M., Gómez Ferro, D. A Critique of snapshot isolation. In *Proceedings of the 7th ACM European Conference on Computer Systems, EuroSys'12*, Bern, Switzerland 2012, 155–168.

Jingyu Zhou, Meng Xu, Bala Namasivayam, Rusty Sears, John Leach, Dave Rosenthal, Xin Dong, Alec Grieser, Yang Liu, Alvin Moore, Xiaoge Su, and Vishesh Yadav ({jingyu_zhou, meng_xu, bnamasivayam, russell_sears, john_leach, daver, xind, agrieser, yliu68, alvinm, xiaoge_su, vishesh_yadav}@apple.com), Apple Inc., Cupertino, CA, USA.

Alexander Shraer (alexander@cockroachlabs.com), Cockroach Labs, New York, NY, USA.

Alex Miller, Evan Tschannen, Steve Atherton, Andrew J. Beamon, and Bhaskar Muppana ({alex.r.miller, evan.tschannen, steve.atherton, aj.beamon, g.muppana}@snowflake.com), Snowflake Inc., San Mateo, CA, USA.

Will Wilson, Ben Collins, and David Scherer ({will.wilson, ben.collins, david.scherer}@antithesis.com), Vienna, VA, USA.

Copyright held by authors/owners. Publication rights licensed to ACM.

ACM Transactions on Evolutionary Learning and Optimization (TELO)

Open for Submissions

Publishes papers at the intersection of optimization and machine learning, making solid contributions to theory, method and applications in the field.



ACM Transactions on Evolutionary Learning and Optimization (TELO) publishes high-quality, original papers in all areas of evolutionary computation and related areas such as population-based methods, Bayesian optimization, or swarm intelligence. We welcome papers that make solid contributions to theory, method and applications. Relevant domains include continuous, combinatorial or multi-objective optimization.

For further information and to submit your manuscript, visit telo.acm.org



Technical Perspective

What Does Provable Security Mean for Cryptographic Schemes?

By David Pointcheval

IN THE EARLY ages of cryptography, cryptographic schemes were considered secure until an attack could break them. But such an attack may take time to be discovered, and it might be too late. In the 1980s, a new paradigm emerged with the notion of *provable security*, with three important steps: the security model, the computational assumptions, and the proof by reduction.

The most important step is the formalization of a security model, which precisely states what security should mean, or alternatively, which attacks one wants to withstand. With public-key encryption, as studied in the following paper, one could simply expect the *one-wayness*, which basically requires that from a ciphertext, without the decryption key, it is difficult to recover the plaintext. But this is a very weak security notion. It has thereafter been improved into *semantic security*, which can be seen as a computational variant of *perfect privacy*: no adversary can recover one bit of information of the plaintext, still from the ciphertext and without the decryption key.

Thus, the security model first makes precise the goal an adversary could try to achieve, and we want to prevent, and this is usually modeled by a security game between the adversary and a challenger. But it also lists the capabilities of the adversary: what kind of information the adversary can legitimately get? In this setting, the encryption key is public, and so the adversary can encrypt any plaintext of its choice, hence the so-called *chosen-plaintext attacks*. In real-life applications, more information may be available: one can send ciphertexts and, from the answer of the receiver, learn whether the ciphertext is valid or not; or detect whether it contains a particular plaintext or not; or ultimately even be able to get the plaintext. The last scenario is called *chosen-*

ciphertext attack where the adversary can ask for the decryption of any ciphertext of its choice to achieve its goal. It is of course precluded to ask for the decryption of the challenge ciphertext, as it would trivially break the one-wayness or the semantic security, but this is the only limitation. A security model thus defines the goals and the means of an adversary.

For a given cryptographic scheme, as perfect security is almost never achievable, one requires a computational assumption: some long-standing problems are famous, such as the integer factoring, the discrete logarithm problem or lattice-based problems. Some of them have a quite simple statement and have been scrutinized for many years. Their intractability is then well-admitted, and a new efficient algorithm to solve them would be of great importance from the algorithmic point of view. Computational assumptions are some well-defined problems that are widely accepted as difficult.

Eventually, one can state and prove the security claim: under the intractability of that hard problem, the cryptographic scheme achieves the security notion. To demonstrate such a claim, one assumes the existence of an efficient adversary against the security notion, and uses it, as a subroutine, in an algorithm to efficiently break the underlying hard problem. The latter cannot exist, hence, by contradiction, neither can the former. This is a *proof by reduction*, regarding complexity theory, by showing a problem (the long-standing problem) reduces to another problem (the security notion): the latter problem is at least as difficult as the former.

The ElGamal public-key encryption scheme, as explored in the paper, is well-known to have secure instantiations in appropriate groups: against chosen-plaintext attacks, it achieves the semantic security under the De-

cisional Diffie-Hellman Problem. Then, one usually claims it as a *secure* public-key encryption scheme. But if the adversary may have additional information than just the public key, security is no longer guaranteed. In real-life, concrete implementations of provably secure schemes may be insecure because of incorrect behaviors: randomness does not contain enough entropy, more information leaks, by legitimate accesses or using side-channel attacks that exploit power-consumption, execution-time, shared cache memory, among others.

The authors perfectly illustrate that provable security is not a certification for *all* the applications nor *all* the implementations: the ElGamal public-key encryption scheme in OpenPGP is vulnerable to several kinds of attacks, and for multiple reasons. First, the standard that describes ElGamal is open to several different and incompatible interpretations, leading to cross-configuration attacks. Second, in a stronger scenario than just the chosen-plaintext setting, a side-channel attack can recover the decryption key.

Provable security guarantees an adversary will unlikely be able to break a *specific* security notion if the scheme is *correctly* implemented and used in *appropriate* situations. It is thus of primary importance to identify the required security notion for the specific application (no need of excessive security, for efficiency concerns), and to have a correct implementation that prevents any unwanted leakage. Standardization bodies should also make all the requirements more precise to avoid any ambiguities that likely introduce weaknesses. 

David Pointcheval is senior researcher at CNRS, head of the crypto team at INRIA, and chair of the ENS computer science department, Paris, France.

Copyright held by author.

On the (In)Security of ElGamal in OpenPGP

By Luca De Feo, Bertram Poettering, and Alessandro Sorniotti

Abstract

Roughly four decades ago, Taher ElGamal put forward what is today one of the most widely known and best understood public key encryption schemes. ElGamal encryption has been used in many different contexts, chiefly among them by the OpenPGP email encryption standard. Despite its simplicity, or perhaps because of it, in reality there is a large degree of ambiguity on several key aspects of the cipher. Each library in the OpenPGP ecosystem seems to have implemented a slightly different “flavor” of ElGamal encryption. While—taken in isolation—each implementation may be secure, we reveal that in the interoperable world of OpenPGP, unforeseen cross-configuration attacks become possible. Concretely, we propose different such attacks and show their practical efficacy by recovering plaintexts and even secret keys.

1. INTRODUCTION

The 1984 ElGamal cryptosystem⁹ is one of the oldest and best-known public key encryption schemes. In the 80s and 90s, it earned wide adoption for being simultaneously efficient and patent-free. Its most prominent use is in OpenPGP,⁶ a standard for consumable and interoperable email security, where it has been the default and most popular encryption option for decades. At the time of writing, still at least one in six registered OpenPGP keys have an ElGamal subkey, with about 1000 new registrations per year.

The ElGamal scheme builds on elegant mathematical structures and can be defined very compactly. This simplicity, together with the opportunity to mature for roughly four decades now, suggests that a crisp specification with clear parameter choices, rules, and algorithms would be present in international standards, in particular in OpenPGP. Surprisingly, this turns out *not* to be the case: Our research reveals that OpenPGP’s understanding of ElGamal encryption is open to interpretation, with several choices subject to the discretion of the implementer.

In this paper, we consider *cross-configuration attacks* on OpenPGP. Such attacks emerge when different interpretations (‘configurations’) of the same standard interact insecurely with each other. Toward identifying such conditions for ElGamal encryption, we need to first understand the universe of OpenPGP interpretations that are used in practice. We approach this challenge from various angles: We carefully study RFC4880⁶ which defines OpenPGP, we inspect the source code of a number of relevant OpenPGP-implementing software libraries and we conduct a large-scale examination of millions of keys registered on OpenPGP key servers.

Our results reveal an insecure posture. For instance, we develop a plaintext recovery attack that can be mounted on ciphertexts produced by the ubiquitous GNU Privacy Guard

against keys generated following the original ElGamal specification.⁹ The attack is effective against 2048-bit keys, a bit length that is considered practically secure today. We further illustrate how cross-configuration attacks can be combined with known side-channel exploitation techniques like FLUSH+RELOAD²³ or PRIME+PROBE.²⁰ Concretely, using the example of `gcrypt`—the cryptographic library underlying the GNU Privacy Guard that has already been fixed twice after seminal work^{14, 23} on side-channel attacks—we demonstrate that if standards conform 2048 bit ElGamal key is used to decrypt a cipher-text, then an attacker that is OS- or VM-colocated with the decrypter might be able to fully recover the decryption key. Given that interoperability is the explicit and almost exclusive goal of any standardization effort, and common-place in the OpenPGP world, we conclude that our cross-configuration attack conditions are as realistic as the attack results in awakening.

This manuscript is organized as follows: In Section 2, we survey (a) the meaningful options available when implementing ElGamal encryption, (b) the options adopted by different cryptographic libraries, and (c) the options picked by over 800,000 users in practice (as far as reflected on key server databases). In Section 3, we recall various standard algorithms for solving discrete logarithms. In Section 4, we describe “vanilla” cross-configuration attacks, and in Section 5, we describe those combined with side-channel attacks. In Section 6, we conduct end-to-end exploits and describe how we bring in the required side-channel information. We conclude in Section 7.

1.1. Related work

Since ElGamal encryption was first proposed, a number of research results revealed security issues, both in implementations (e.g., CVE-2018-6829, CVE-2018-6594) and at the specification level. For instance, Bone *et al.*⁵ identified security conditions if ElGamal encryption is used in textbook form. However, as OpenPGP employs ElGamal exclusively for key transport, the attacks do not seem to apply in practice. To the best of our knowledge, our work is the first to challenge the security of ElGamal encryption in the specific way OpenPGP builds and relies on it.

Protecting modular exponentiation from side-channel attacks has been the subject of research for years.^{2, 10, 11, 15} Specifically, cache-based side channels have been explored

The original version of this paper was published in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, Nov. 2021, 2066–2080; <https://doi.org/10.1145/3460120.3485257>

extensively, using various techniques such as EVICT+TIME,⁴ PRIME+PROBE,²⁰ or FLUSH+RELOAD.^{3, 23} Yarom and Falkner²³ in particular target the `gcrypto` implementation of modular exponentiation which at the time was using square-and-multiply. Prompted by that work, the implementation was significantly revised. The changes are however not sufficient to protect against the attack we present in Section 5. Most cache-based side-channel attacks target L1 or L2. However, Liu *et al.*¹⁴ show that exploitation based on last-level cache is possible for attacks targeting both data and instructions. The work also targets the `gcrypto` implementation of modular exponentiation and the library has also been fixed to avoid leaking table accesses. Once again, however, this is not sufficient to prevent the attack described in Section 5.

2. ELGAMAL ENCRYPTION

After providing an abstract formulation of ElGamal encryption, we present common ways to instantiate the scheme in practice.

Let G be a group and $g \in G$ a generator. To create a key pair (sk, pk) , pick a random integer x , compute the element $X = g^x$, and output $(sk, pk) := (x, X)$. Given pk , to encrypt a message M , pick an ephemeral random integer y , compute the elements $Y = g^y$ and $Z = X^y = g^{xy}$, and output $C = (C_1, C_2) := (Y, M \cdot Z)$ as the ciphertext. Given sk , to decrypt C , first recover element Z from C_1 as per $Z = Y^x = g^{yx}$ and then use C_2, Z to recover $M = C_2/Z$.

To instantiate the scheme, the following details have to be fixed: Which group G shall be used? How is generator g chosen? Shall g generate the full group G or just a subgroup? From which sets and according to which distributions are exponents x, y picked? Multiple configurations for the parameters are possible and promise to lead to correct and secure public key encryption instances. Below we describe four such configurations that have in common that G is a “prime field group,” that is, the multiplicative group $\mathbb{Z}_p^* = (\mathbb{Z}/p\mathbb{Z})^*$ of a field $\mathbb{Z}/p\mathbb{Z}$ where p is a large prime number (also referred to as the modulus). In such cases the order of G is given by $\text{ord}(G) = p - 1$ and the order of any subgroup $G' \subseteq G$ is an integer divisor of $p - 1$.

Configuration A: “The original”. The original proposal⁹ demands choosing g such that it generates $p - 1$ elements, that is, the full group. Exponents x, y are picked uniformly from $[1 \dots p - 1]$. The only condition on p is that $p - 1$ has at least one large prime factor [This is to counter the Pohlig–Hellman attack on Discrete Logarithm Problem (DLP).]

If Configuration A is used, any element in G could become a public key. However, some of these public keys would have a lower multiplicative order than others, that is, generate a smaller subgroup, and thus promise less security. (E.g., in the extreme case of $X = 1$, the encryption operation does effectively nothing.) This can be remedied by restricting the ElGamal group operations to a prime order subgroup $G' \subseteq G$. Indeed, if $G' = \langle g \rangle$ has prime order, then all its elements necessarily have the same order (with the one exception of the neutral element which is easily tested for). Let hence $p - 1 = q_0 \cdots q_n$ be the prime factor decomposition of

$\text{ord}(G)$. As p is a large prime number and hence odd, we know that one of these prime factors is 2, and we hence w.l.o.g. write $p - 1 = 2q_1 \cdots q_n$. Now, for any prime $q = q_i$ in this list of factors there exists a subgroup $G' \subseteq G$ of order q . The idea of using such a group G' in cryptography goes back to Schnorr.¹⁸

Configuration B: “ElGamal over Schnorr groups”. Pick a large prime group order q and a prime p such that $q \mid (p - 1)$, choose a generator γ with $\langle \gamma \rangle = G$ and let $g = \gamma^{(p-1)/q}$ and $G' = \langle g \rangle$. Note that this implies $\text{ord}(G') = q$. Pick exponents x, y in $[1 \dots q - 1]$.

This setup satisfies the Pohlig–Hellman condition and ensures that all non-trivial elements of G' have the same order. Further, if $q \ll p$, as exponents x, y are now picked from $[1 \dots q - 1]$ instead of $[1 \dots p - 1]$, implementations of Configuration B are very efficient.

While Configuration B removes the risk of accidentally relying on small order elements, an adversary can still exploit the existence of small subgroups in an active attack: Observe that for any t with $t \mid (p - 1)$ there exists an element α of order t , that is, with $\alpha^t = 1$ and $1 \notin \{\alpha, \alpha^2, \dots, \alpha^{t-1}\}$. Fix such a pair (t, α) , let $C = (C_1, C_2)$ be an honestly generated ciphertext as above, and assume an adversary intercepts C and re-injects it as $C' = (C'_1, C'_2) := (\alpha C_1, \alpha^h C_2)$, for $h \in [1 \dots t]$. When processing C' , the decrypter obtains $M' = C'_2 / Z' = (\alpha^h C_2) / (\alpha^x C_1^x) = \alpha^{h-x} M$. Note that $M' = M$ iff $\alpha^{h-x} = 1$ iff $h \equiv x \pmod{t}$; otherwise $M' \neq M$. That is, by testing whether the injected ciphertext C' properly decrypts, the adversary can confirm or refute a guess on $x \pmod{t}$. Recovering $x \pmod{t_1}, x \pmod{t_2}, \dots$ for carefully picked $t_1, t_2, \dots$ allows recovering the full exponent x via the Chinese Remainder Theorem. As for each t -value the adversary has to request up to t decryptions, small t -values are a precondition for the attack to be effective. So is the existence of sufficiently many suitable t -values. Hence, one countermeasure against the attack is to ensure that the number of distinct divisors t of $p - 1$ is very small, and another is to arrange that all admissible t -values are very large. Configurations C and D formalize these ideas and can be traced back to Pollard¹⁷ and Lim and Lee.¹³

Configuration C: “ElGamal over safe primes”. The number of subgroups is minimized: Pick a large prime group order q and a prime modulus p such that $p - 1 = 2q$, choose a generator γ with $\langle \gamma \rangle = G$ and let $g = \gamma^{(p-1)/q} = \gamma^2$ and $G' = \langle g \rangle$. Pick exponents x, y in $[1 \dots q - 1]$. Note that $p \approx q$. Choosing $g = 4 = (\pm 2)^2$ is always feasible, leading to smaller parameters.

Configuration D: “ElGamal over Lim–Lee primes”. All non-trivial subgroups are large: Pick a large prime modulus p such that for the prime factorization $p - 1 = 2q_1 \cdots q_n$ all q_i have a similar, large bit length. (Breaking DLP is then hard for all group orders q_i .) Choose a generator γ with $\langle \gamma \rangle = G$ and let $g = \gamma^{(p-1)/q}$ and $G' = \langle g \rangle$. Pick exponents x, y in $[1 \dots q - 1]$.

As exponents in Configuration C became large again, some implementations work with “short exponents” for improved performance. Configuration D maintains good performance without such tricks. However, its parameters are larger. When building a secure system from scratch, it seems advisable to use one of these two options.

2.1. ElGamal in OpenPGP

The standard. A widely-deployed cryptography standard that suggests using, and mandates implementing, ElGamal encryption is OpenPGP. While its first version was put forward in 1998, the latest official version appeared in 2007 as RFC4880.⁶ We observe that the RFC refers to external documents for the specification of the ElGamal cryptosystem. Precisely, Section 9.1 provides two references that both ultimately resolve to ElGamal's original paper,⁹ that is, our "Configuration A". We note that Sections 5.1 and 5.5, which define how ElGamal ciphertexts, public keys, and secret keys are to be represented as binary strings, add no further conditions on the generation and use of ElGamal keys and ciphertexts.

OpenPGP libraries. We inspected how OpenPGP libraries interpret the standard. Here are our findings on three popular ones:

`Go` `Go` does not provide code to generate ElGamal keys; it only provides algorithms to encrypt and decrypt. The ephemeral exponent y used in encryption is chosen from $[0 \dots p-1]$. This fully conforms with RFC4880.

`Crypto++` The `Crypto++` library follows Configuration C by generating a random safe prime and choosing for the generator the smallest quadratic residue. It though deviates from Configuration C by picking both x and y from a *short interval* of size roughly twice the security parameter. (See the full version^{7,8} for details.) As the group generator is not primitive, this setting conflicts with RFC4880.

`gcrypt` The `gcrypt` library generates a Lim-Lee modulus like in Configuration D; the minimum size of the prime factors $q_i \mid (p-1)$ is roughly twice the security parameter. (See the full version^{7,8} for details.) Unlike in Configuration D, the generator is chosen to be the smallest integer generating the full group $\mathbb{Z}_p^\times$. Both exponents x and y are sampled from *short intervals* of size roughly $q^{3/2}$. Due to the short exponents, this setting conflicts with RFC4880.

ElGamal parameters in the wild. To get a complete picture of the use of ElGamal in OpenPGP, it is not sufficient to look at the prominent open-source libraries. Indeed a large portion of the user base relies on proprietary or exotic implementations, which are impossible to track. To address these, we look at the OpenPGP keys registered on public key servers. We analyze an OpenPGP server dump¹ produced on Jan 15, 2021 subkeys.

An OpenPGP ElGamal public key consists of the triplet (p, g, X) . This information alone is not sufficient to ascribe the key to one configuration with certainty, however, partial information can be deduced by attempting to factor $p-1$. For example, safe primes are easily recognized by running a primality test on $(p-1)/2$, then a quadratic residuosity test reveals whether g generates the prime order subgroup or the full group. For random primes of the sizes we look at, it is in general infeasible to obtain a full factorization of $p-1$, however, partial factorizations and residuosity tests let us at least formulate credible hypotheses on the key generation process.

To classify public keys, we conducted trial division on $p-1$ with primes up to 2^{25} , then repeatedly applied the elliptic curve factorization method (ECM),¹² until we felt guilty for the carbon emissions. Then we applied n -th residuosity tests to g for the factors we found. We did not attempt to gain information on the exponent x that defines $X = g^x$. In our findings, 69.4% of the keys use safe primes, 25.3% use moduli that were very likely generated using the Lim-Lee method, and 5.0% use what we call "quasi-safe primes", that is, primes of the form $p = Q \cdot q + 1$ where q is an unusually large prime (0.988 times the size of p on average) and $Q > 2$. Finally, there are only 2158 moduli (less than 0.4%) for which we found non-trivial factors, but we were not able to finish the factorization, indicating that they were either chosen at random or like in Configuration B. These would almost be irrelevant if it was not for the attack we describe in Section 4.

3. COMPUTING DISCRETE LOGARITHMS

In the next sections, we will need to solve discrete logarithms given partial knowledge of the exponent. We review here the necessary algorithms. In what follows, let g be a group generator of order N , and let $X = g^x$ be the element of which we seek the discrete logarithm.

Pollard's Rho algorithm¹⁷ is the most efficient generic algorithm to compute discrete logarithms. On average, it performs $\sqrt{\pi N/2}$ group operations and uses a constant amount of memory. Pollard¹⁷ also introduced the lesser-known Lambda method, which performs better when it is known that $x < B \ll N$, requiring only $\approx 2\sqrt{B}$ group operations and $O(\log(B))$ memory (Section 5.1 of van Oorschot and Wiener²²).

When $N = LM$, we can compute $x \bmod L$ by solving a discrete logarithm in the group of order L generated by g^M . The Pohlig-Hellman algorithm¹⁶ applies this to all prime factors of N , and then recovers x via the Chinese Remainder Theorem (CRT).

We will combine all of these techniques in the case where $N = q_0 \cdots q_n$, and $x < B$. Assume $q_0 < \cdots < q_n$, and let $Q = q_0 \cdots q_{n-1}$. We first compute $x \bmod q_i$ for $0 \leq i < n$ using Pollard's Rho, then use the CRT to compute $w := x \bmod Q$. Thus $g^x = g^{w+Qz}$ for some unknown $z < \lceil B/Q \rceil$. Finally, we recover z as the discrete logarithm of $g^{x/w}/g^w$ to base g^Q , using Pollard's Lambda. The total cost is $O(\sqrt{q_{n-1}} + \sqrt{B/Q})$ time and $O(\log(B/Q))$ storage. We stress that this strategy is only better than Pohlig-Hellman when $B \ll N$. It was proposed by van Oorschot and Wiener to reduce the security of variants of Configuration A that use short exponents for public keys.²¹ We will use it, instead, to recover ephemeral secrets in cross-configuration scenarios described in Section 4.

In Section 5, we will need to solve discrete logarithm instances where some non-adjacent bits of x are known. Neither the Rho nor the Lambda method can take advantage of this information, but the simpler baby-step/giant-step (BSGS) method¹⁹ can. If x has n unknown bits, BSGS performs $1.5 \cdot 2^{n/2}$ group operations on average, and stores $2^{n/2}$ hash table entries. A linear time/memory trade-off is possible in BSGS.

However, as n becomes larger, BSGS has two important drawbacks: it uses unrealistically large amounts of memory, and it parallelizes poorly. A better alternative is van Oorschot

and Wiener's (vOW) parallel collision search applied to meet-in-the-middle algorithms (Section 5.3 of van Oorschot and Wiener²²), which is much more memory efficient, and promises a linear parallel speed-up. Based on their analysis, vOW is expected to require $7 \cdot 2^{3n/4-m/2-1}n$ group operations, where 2^m is the amount of storage available, counted as a number of hash table entries, and subject to the constraint $m \leq n/2$.

4. CROSS-CONFIGURATION ATTACKS

The disagreements on the interpretation of the OpenPGP standard may raise doubts about the interoperability between the libraries. For instance, in an imaginary setting where the `Crypto++` code is used to generate a key pair, the `Go` code is used to encrypt a message to the public key, and the `gcrypt` code is used to decrypt the ciphertext, it has to be asked whether confidentiality is maintained. While in a basic scenario these three libraries can, to the best of our knowledge, in fact, interoperate securely, we shall now see that some choices made by `Crypto++` and `gcrypt` prove to be fatal in a broader context.

As van Oorschot and Wiener²¹ have already observed: if (1) $p-1$ contains enough small factors, and (2) g generates the full group (e.g., in Configuration A), then using short exponents in the public key $X = g^x$ can lead to key recovery, as described in Section 3. As we have seen, `Crypto++` uses safe primes, so it is not at risk. Although `gcrypt` generates $p-1$ with many distinct factors, none of these is small, and it is thus also safe.

However, both `Crypto++` and `gcrypt` also use short exponents in the ephemeral key $Y = g^y$. But the generator g is part of a public key that may have been generated by a different library, possibly one adhering to Configuration A. Our analysis of registered keys shows that such public keys, albeit rare, do exist. This immediately leads to a message recovery attack: given a ciphertext $(g^y, M \cdot g^{xy})$, the attacker finds y , uses the public key to compute g^{xy} , and recovers the cleartext M .

To recap, whenever: (1) the public key of the receiver defines a generator g of a group containing “enough” small-order subgroups, and (2) the sender uses short exponents in the ephemeral key, an attacker who intercepts messages can decrypt any communication from sender to receiver. The exact cost of the attack depends on the number and size of the small subgroups, which we analyze next.

4.1. Practical exploitation

Like in Section 3, let $N = q_0 \cdots q_n =: Qq_n$ be the group order of a public key, with q_n not necessarily prime. Assume that ephemeral exponents for encryption are sampled from a short interval $[0 \dots B]$, with $B \ll N$. As we saw, the cost of recovering the exponent depends on two factors: the largest divisor q_{n-1} of Q , and B/Q (note that, while Q and the q_i 's depend on the receiver's public key, the bound B is set by the sender).

To quantify the threat, we look at these parameters for those groups in the OpenPGP key dump that contain small factors. None of the “quasi-safe” moduli are seriously at risk, since Q is too small. From the remaining 2158 keys, we narrow down the selection to those that had 2048 bits modulus, were not created before 2016, and were not expired nor

revoked, leaving us with 2071 keys. In Figure 1, we plot the (normalized) cumulative counting function

$$\Phi(\lambda, \rho) = \#\{N = q_0 \cdots q_n = Qq_n \mid q_0, \dots, q_{n-1} \leq 2^\rho, Q \geq 2^\lambda\},$$

which counts, among the 2071 keys, how many are exposed to message recovery with a Rho effort at most $2^{\rho/2} \sqrt{\pi}/2$ and a Lambda effort at most $\sqrt{B}/2^{\lambda/2-1}$. Note that the factors of a single group order can be arranged in more than one way, and thus appear in more than one bucket.

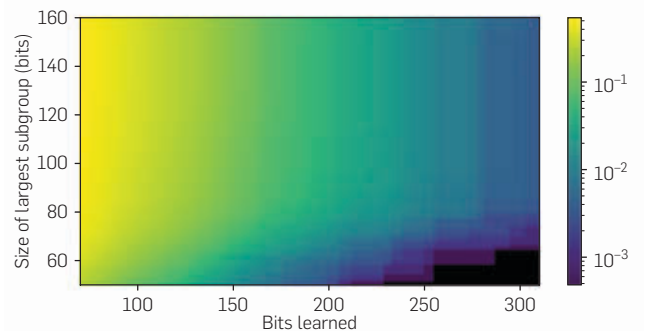
From $\phi(\lambda, \rho)$, we obtain the number of keys for which message recovery is possible with a given effort with respect to a given encrypting library. For example, for 2048-bit moduli `gcrypt` implements the sampling of y such that $y < 2^{344}$: limiting the Lambda and Rho efforts to roughly 2^{40} modular multiplications, a computation well within reach of a casual attacker, we find that there are at least $\phi(344-80, 80) = 15$ vulnerable keys. Even more concerning is the case of `Crypto++`, which samples exponents in $[1 \dots 2^{226}]$: in this case, the number of weak keys goes up to $\phi(226-80, 80) \geq 231$. It is plausible that a state-level attacker could target as many as $\phi(344-140, 140) \geq 83$ (`gcrypt`) and $\phi(226-140, 140) \geq 895$ (`Crypto++`) keys.

To confirm the vulnerability we used GPG to encrypt a message to the only key contributing to $\phi(344-56, 59)$, expecting to complete the attack using roughly 2^{30} modular multiplications. We were able to recover the plaintext in less than 2.5 hours on a single Intel E5-2640 core clocked at 2.40GHz.

5. ATTACKS ENABLED BY SIDE-CHANNELS

The attack conditions exploited in Section 4 did not depend on implementation flaws but were solely implied by different implementers coming up with different conceptual interpretations of ElGamal encryption. The current section considers cross-configuration attacks that are also based on implementation weaknesses. We focus specifically on runtime leakage of the modular exponentiation algorithm of `gcrypt`, which represents a crucial component of its ElGamal implementation. Specifically, an exponentiation algorithm (EA) computes $R = B^x$ from B and x , where base B is an element of a cyclic group and exponent x is a non-negative integer. While the `gcrypt` implementation took explicit measures to prevent side-channel leakage (in

Figure 1. Normalized cumulative distribution function $\Phi(\lambda, \rho)$ of group orders N that contain a 2^ρ -smooth factor Q of at least λ bits.



particular, to protect exponent x), by exploiting a condition that the authors allegedly overlooked we present a key recovery attack that significantly reduces the security margin of `gcrypt`. In isolation, that is, not in the cross-configuration setting, the attack is only practically exploitable against `gcrypt`-generated keys with moduli up to 1024 bits. However, we show that the vulnerability is more worrisome in an interoperability context where `gcrypt` decryption is used in combination with a `Crypto++` key: in this case, we experimentally confirm a practical key recovery attack against a non-negligible fraction of `Crypto++` keys with 2048 bits modulus. In the full version,^{7,8} we provide similar analyses of the EAs of the `Go` and `Crypto++` libraries.

The `gcrypt` library implements its EA via the sliding-window method, which is derived from the basic Square-and-Multiply algorithm. The algorithm depends on a parameter w (for “width” or “window length”) that is instantiated such that $2 \leq w \leq 5$. The first step of the EA is independent of exponent x and tabulates the values $B^1, B^3, \dots, B^{2^w-1}$. The second step is independent of base B and regroups the exponent bits into an *odd-digit representation* (ODR). Precisely, an ODR of a non-negative integer x is a sequence $\Gamma = (\gamma_0, \dots, \gamma_o)$ such that

$$x = \sum_{j=0}^L 2^j \gamma_j \quad \text{and} \quad \forall j: \gamma_j \in \{0\} \cup \{1, 3, \dots, 2^w - 1\}.$$

The exponent x is first converted into ODR form that has leading coefficient $\gamma_0 = 1$,^a the ODR is then encoded into a sequence $(e_0, c_1, e_1, c_2, e_2, \dots, c_L, e_L, c_{L+1})$ such that its odd-index subsequence $(e_0, e_1, \dots, e_L)$ coincides with the support $(\gamma_j)_{j:\gamma_j \neq 0}$ of Γ (i.e., the set of non-zero elements; note here we have $e_0 = \gamma_0 = 1$, and for all e_i we have $e_i \in \{1, 3, \dots, 2^w - 1\}$), and the values $c_1, c_2, \dots, c_{L+1}$ correspond with the lengths of the vanishing subsequences of Γ (i.e., the contiguous runs of zero elements). Precisely, the encoding is such that

$$x = \sum_{i=0}^L 2^{\bar{c}_i} e_i \quad \text{where} \quad \forall i: \bar{c}_i = c_{i+1} + \dots + c_{L+1} \quad (1)$$

The third step of the EA is a computation-intensive online phase in which the results of the first two steps are combined.

We provide a pseudo-code version of `gcrypt`’s EA in Figure 2. The function f_w invoked in line 2 (and specified in the full version^{7,8}) computes the exponent encoding $(e_0, c_1, e_1, c_2, e_2, \dots, c_L, e_L, c_{L+1})$ as follows: it sets $e_0 = 1$, then alternates between (a) advancing through the bit sequence of x (from most to least significant bit) until it encounters a bit set to 1, and (b) greedily outputting the largest possible pair (c_i, e_i) subject to $e_i \in \{1, 3, \dots, 2^w - 1\}$. Also lines 5–8 might require further explanation: The code is functionally equivalent with ‘For $j \leftarrow 1$ to c_i : $R \leftarrow R \cdot R$ ’ followed by ‘ $R \leftarrow R \cdot T[e_i]$ ’, but the implementation reduces side-channel leakage by hiding for each multiplication whether the factor W is R or $T[e_i]$, and, in the latter case, which index e_i is used for the table look-up.^b

Despite its built-in protection measures, we identify a side-channel condition in `gcrypt`’s EA that can lead

Figure 2. Core of exponentiation routine of `gcrypt`.

Exponentiation $R \leftarrow B^x$

Parameterized by: w

Registers: $T[1], T[3], \dots, T[2^w - 1], W$

```

1 For odd  $0 \leq i < 2^w$ :  $T[i] \leftarrow B^i$  // Initialize  $T[\dots]$ 
2  $(e_0, c_1, e_1, \dots, c_L, e_L, c_{L+1}) \leftarrow f_w(x)$  // Encode exponent
3  $R \leftarrow B$  // Implicitly process  $e_0 = 1$ 
4 For  $i \leftarrow 1$  to  $L$ : // Main loop
5 // Square  $c_i$  times, then multiply with  $T[e_i]$ 
6 For  $j \leftarrow 1$  to  $c_i + 1$ :
7 Securely  $W \leftarrow [j \leq c_i ? R : T[e_i]]$ 
8  $R \leftarrow R \cdot W$ 
9 // Square  $c_{L+1}$  times
10 For  $j \leftarrow 1$  to  $c_{L+1}$ :
11  $R \leftarrow R \cdot R$ 
12 Return  $R$ 
```

to full exponent recovery. The root of the problem is that the algorithm implements sliding-window exponentiation. Intuitively, this method covers the digits of the bit-representation of the exponent x with a sequence of w -wide non-overlapping windows such that every 1-bit of the exponent is covered by a window and conditioned on this the number of windows is minimized. In Equation (1), value L corresponds with the number of windows used for this, any coefficient $\bar{c}_i$ corresponds with the position of the i -th window, and any coefficient e_i encodes the w exponent bits that the i -th window covers. In the EA implementation, processing a window corresponds with a multiplication $R \leftarrow R \cdot T[e_i]$ while bridging a gap between two windows corresponds with a sequence of squaring operations $R \leftarrow R \cdot R$. These operations are jointly implemented in lines 7,8, and each iteration of the loop of line 4 processes one gap-window pair.

The approach of our side-channel attack is to closely monitor the execution of line 4: The number of iterations of the loop during one exponentiation immediately reveals the encoding length L , and the time taken for the i -th iteration is, by line 6, linear in $c_i + 1$ and thus leaks, one by one, the coefficients $c_1, \dots, c_L$.^c Finally, we can recover the coefficient c_{L+1} by similarly monitoring the loop of line 10. That is, if $(e_0, c_1, e_1, \dots, c_L, e_L, c_{L+1}) = f_w(x)$ is the encoding of exponent x then our measurements reveal all the c_i components while, for now, the e_i components remain hidden.

Before assessing the options for amplifying the extracted information toward recovering the full exponent, we make a final observation on `gcrypt`’s EA. Recall that line 7 tries to hide both the table index e_i and whether the multiplication in line 8 is with $T[e_i]$ or R . However, as our method recovers the coefficients c_i straightaway, and exclusively the last iteration of the loop of line 6 will perform multiplication with $W = T[e_i]$ (rather than with $W = R$), line 7 *de facto* just hides the e_i coefficients. Thus, replacing lines 6–8 with the instructions “For $j \leftarrow 1$ to c_i : $R \leftarrow R \cdot R$ ” followed by “Securely $R \leftarrow R \cdot T[e_i]$ ”

^a Every $x \geq 1$ has at least one, but often multiple, ODRs with leading coefficient 1.

^b Precisely, line 7 implements the instruction “If $j \leq c_i$ then $W \leftarrow R$; else, if $j = c_i + 1$, then $W \leftarrow T[e_i]$ ” but ensures it neither leaks which if-branch is taken nor what the value of e_i is.

^c This argument assumes multiplications and squarings require uniform time. This is the case for the `gcrypt` routines.

would result in code that is equivalent from a security perspective yet simpler and more efficient.^d We, thus, conclude that the authors of `gcrypt` were likely not aware of the fact that the coefficients $c_1, \dots, c_{L+1}$ leak so easily.

Now that we retrieve copies of the coefficients $c_1, \dots, c_{L+1}$, the next subsections are dedicated to assessing the value of this information toward an attack that recovers the full exponent.

5.1. Modeling the leakage

As a warm-up, observe that when $w = 1$ the coefficients c_i leak the whole exponent, indeed all e_i are necessarily equal to 1 in this case. For larger values of w , however, a search over the possible values of e_i is necessary. Our goal here is to quantify this partial leakage.

Letting $n+1 = \lceil \log_2(x+1) \rceil$ be the bit length of x , we start by observing that $n = \sum c_i$ and is thus fully known. The sequence of c_i 's indicates the position of each e_i in the binary writing of x , and from the description of f_w we know that e_i is odd and $e_i < \min(2^w, 2^e)$, leaving us with $\min(w, c_i) - 1$ unknown bits in e_i . Note that there is no e_{L+1} associated to c_{L+1} , which indeed leaks the number of trailing zeros of x . Thus, the total number of unknown bits of x , which we shall call the *entropy* of x , is exactly

$$H_w(x) = \sum_{i=1}^L (\min(c_i, w) - 1) \quad (2)$$

Given h , h^x and the leakage for x , the computational effort for finding x grows exponentially with $H_w(x)$. The value $H_w(x)$ depends on x , on the window size w , and on the algorithm f_w used for the encoding. We are thus interested in estimating statistics on $H_w(x)$ for secrets x of n bits.

To this end, we model the encoding function f_w as a Markov process: Model x as a continuous stream of independent and uniformly distributed bits, for any fixed w the leakage of the encoding f_w is described by a finite state machine (see the full version^{7,8} for a detailed description), which we then convert to a Markov chain.

Using the Markov chain central limit theorem, we compute that as $n \rightarrow \infty$ the entropy $H_w(x)$ tends to a normal variate of mean $n\mu$ and variance $n\sigma^2$, where μ and σ^2 are the “per bit” mean and variance tabulated in Table 1. For example, for $w = 2$, the average per-bit entropy is $\mu = 7/24$, meaning that, given a uniformly sampled $n+1$ -bits exponent and its leakage $c_1, \dots, c_{L+1}$, on average $7n/24$ bits remain to be found.

Of course, these asymptotic approximations only give a rough estimate of the actual probability distribution of $H_w(x)$ for a given bit length n , however, in our experiments we found them to be quite accurate in practice even for small values of n . Using these approximations, we will give rather precise estimates of the computational effort needed to recover random secret keys generated by `gcrypt` and other libraries.

5.2. Key recovery

To assess the practical impact of the leakage of sliding window exponentiation exposed so far we need to look into the

^d The performance gain comes from the possibility of removing some of the decoy code required for securely implementing line 7. For further details, see the source code.

exact parameters used by the various libraries. Recall that `gcrypt` uses short secret exponents. The window size used for exponentiation goes from 1 to 5, depending on the bit size of the exponent. Table 2 summarizes the parameters adopted by `gcrypt` for some common modulus sizes.

Based on `gcrypt`'s parameters, Table 2 also reports on the mean leakage entropy $H_w(x)$ as deduced from Table 1. We can thus compare the expected difficulty of recovering the secret exponent via the leakage against the target difficulty of solving discrete logarithms directly. `gcrypt`'s moduli are parameterized so that the best algorithm for computing discrete logarithms is, indifferently, index calculus in $\mathbb{Z}_p$, or Pollard's Rho in the subgroup of largest prime order q . For simplicity, we use the cost of Pollard's Rho, namely $\sqrt{\pi q}/2$, as our reference difficulty.^e

To exploit the leakage, we resort to vOW parallel collision search (see Section 3), which costs $7 \cdot 2^{3H_w(x)/4 - m/2 - 1} H_w(x)$ modular multiplications, where 2^m is the amount of RAM available, counted as a number of hash table entries (e.g., for $|p| = 2048$ an entry occupies slightly more than 256 bytes). Both Pollard's Rho and vOW can be parallelized with a linear speed-up (Section 5.1–3 of van Oorschot and Wiener²²), justifying the comparison.

The last two columns of Table 2 report the base 2 logarithm of the expected efforts of Pollard's Rho, and of vOW assuming a hash table with at most 2^{60} entries (note that even this “little” memory is quite unrealistic). Comparing the columns, the leakage of windowed exponentiation does not appear to be a serious threat for the average `gcrypt` secret key. However this assessment is deceiving on two accounts: (a) the standard deviation of $H_w(x)$ is rather large, and thus some keys will be much easier to break than others; and (b)

^e Since in practice q is unknown to an attacker, this ignores the cost of factoring $p - 1$ first, but accounting for it would only make the side channel attack look better.

Table 1. Per bit statistics of the entropy $H_w(x)$ as $n \rightarrow \infty$ for different values of the window width w .

window width w	μ	σ^2
2	7/24	0.058449
3	7/16	0.098632
4	341/640	0.126731
5	925/1536	0.145411

Table 2. Group ($|q|$) and secret exponent ($|x|$) bit sizes, and window size (w) used by `gcrypt` for each modulus size ($|p|$). $E(H_w(x))$ is the mean leakage entropy estimated using Table 1. “Pollard” indicates the (base 2 log of the) expected running time of Pollard's Rho algorithm in a group of size q , as a number of modular multiplications. “vOW” indicates the expected running time of van Oorschot and Wiener's algorithm using a table of 2^{60} entries.

$ p $	$ q $	$ x $	w	$E(H_w(x))$	Pollard	vOW
1024	166	250	3	108.94	82.65	63.04
2048	226	340	4	180.62	113.15	114.77
3072	270	406	4	215.79	134.65	141.40
4096	306	460	4	244.56	152.65	163.16

`gcrypt` is assumed to be secure even when it operates on secret keys generated by a different library, as long as these are valid OpenPGP keys. Some libraries, such as `Crypto++`, generate secret exponents with as few as $|q|$ bits, their keys are thus easier targets.

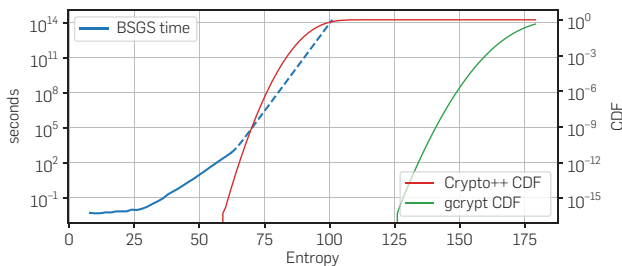
Taking for reference the most commonly used modulus size $|p| = 2048$, `Crypto++` exponents are between 1 and 226 bits long, and the window size used by `gcrypt` is at most $w = 3$, leading to an average entropy of 98.0 bits. Worse still, more than one in 10,000 keys is expected to have at most 80 bits of leakage entropy: for such a small search space, `vOW` is expected to only require 2^{50} modular multiplications (roughly 2^{35} Gcycles in software) for a hash table of 2^{35} entries (roughly 4 TiB). We thus conclude that attacking a non-negligible proportion of `Crypto++`-generated public keys within `gcrypt`'s decryption routine is well within reach of a moderately resourceful adversary.^f

5.3. Proof of concept

To confirm the reality of the threat, without going as far as spending several thousands of dollars, we implement a simple parallelized BSGS using GMP 6.1.2, and test it on a node equipped with 20 Xeon E5-2640 cores clocked at 2.40GHz and 64GiB of RAM. Due to memory contention, the parallel speed-up is sublinear, nevertheless, we observe a speed-up slightly above 10× using all 20 cores, thus justifying our preference for BSGS over the more complex `vOW` algorithm which is marred by larger constants. Our implementation uses hash table entries of 16 bytes, independently of the modulus size; the node can thus accommodate for tables with as many as 2^{32} entries, ensuring the running time scales like the square root of the entropy, up to 64 bits of entropy. Figure 3 plots the running time of BSGS for increasing values of $H_w(x)$, and superimposes the cumulative distribution function of $H_w(x)$ for `Crypto++` and `gcrypt`, indicating what percentage of keys is broken in a given time. Since BSGS is deterministic, we use for the benchmarks the exponent that is tested by BSGS last, thus the expected running time for an average exponent of given entropy is half the time reported in Figure 3.

^f At the time of writing, 8 Amazon EC2 `r6g.metal` instances with 0.5 TiB of RAM and 64 virtual CPUs each cost about 12,000\$ per month.

Figure 3. Running time of BSGS for increasing values of the entropy $H_w(x)$, on our node equipped with 20 cores and 64GiB of RAM. The dashed line indicates extrapolated running times. Also represented the cumulative distribution functions of the normal distributions of parameters $(225 \cdot 7/16, 225 \cdot 0.098632)$ and $(339 \cdot 341/640, 339 \cdot 0.126731)$, approximately corresponding to the entropy distributions of `Crypto++` and `gcrypt` for a modulus of 2,048 bits.



As a final confirmation, we instrument the `Crypto++` code to generate random secret exponents in a tight loop, and output them along with their leakage entropy. Over 2^{28} samples, we obtained a 226 bits exponent with 62 bits of leakage entropy. Our probing strategy described in Section 6 produces the expected leakage pattern, from which our BSGS code recovers the exponent in approximately 30 min using 20 cores.

6. PRACTICAL EXPLOITATION

In this section, we prototype the attacks described in Section 5 to show that practical exploitation is possible. The attacks are mounted on a Core i7-8650U CPU with Ubuntu 18.04.5. We demonstrate how FLUSH+RELOAD may be employed to extract leakage from the sliding window modular exponentiation implementation of `gcrypt` and use that to obtain full key recovery. The exploitation of `gcrypt` presents interesting aspects for two reasons: (i) the library was patched to protect its modular exponentiation routine against cache-based side channels attacks highlighted by Yarom and Falkner²³ and by Liu *et al.*¹⁴; and (ii) the attack can be conducted on commodity hardware specifically owing to the ElGamal interoperability issues highlighted in Section 2.

6.1. Threat model

We consider a victim process that uses a private key to decrypt ElGamal ciphertexts. We assume the attacker has knowledge of the victim program and of the specific decryption algorithm that is used. We also assume that the attacker may cause the victim process to execute, for example, because the victim uses a PGP-enabled mail client that automatically decrypts emails upon reception. Consequently, we assume that the attacker is able to synchronize side channel measurements with the victim execution. No physical access is required to perform the attacks. Given that we target cache-side channels, the attacker may either be local, or it may be running on a different, colocated, VM. The former may target any cache level, the latter may only target LLC (unless the VM is hyperthread-colocated). Wlog we conduct the exploitation against the L1 cache: as shown by Liu *et al.*,¹⁴ the same exploitation may be achieved against higher cache levels.

6.2. Environment

We target the function `_gcry_mpi_powm` of the latest version `gcrypt` shipped by Ubuntu 18.04.5 LTS. The function implements the approach described in Figure 2. The implementation takes a few precautions to avoid leaking sensitive data and instruction cache accesses. In particular, it hides (i) whether the operation at line 8 is a modular multiplication or squaring; and (ii) whether, at line 7, W is initialized with R or with an entry from table T (and in that case, it also hides which position e_i of the table is loaded). This is accomplished by way of a conditional `memcpy`-like operation (`mpi_set_cond`) that—depending on whether a flag argument is one or zero—copies a source operand into a destination operand, or leaves the destination operand unchanged. Masking is employed to ensure that an attacker cannot learn

the nature of the operation by observing the control flow or changes to data or instruction cache. Furthermore, the flag is set using branchless operations.

Despite these precautions, considerable leakage can still be obtained by an attacker who can observe control flow-dependent cache perturbations: in particular, the full set of values $c_1, \dots, c_L$ can be recovered by counting the number of iterations of the `for` loop at line 6 for each of the iterations of the loop at line 4. The attacker may obtain this information by monitoring the state of the instruction cache line that contains code that is executed as part of an iteration of the outer loop *and not* of the inner loop. In practice, this can be achieved with the FLUSH+RELOAD side channel applied on a line of the instruction cache that is executed once per outer loop before the start of the inner one. The probe reveals when such a cache line is executed, and the time between two probes would depend on the execution time of the inner `for` loop. Crucially, the latter depends on the set of values $c_1, \dots, c_L$, which would thus be leaked. This strategy is applicable to the `_gcry_mpi_powm` function: the implementation inlines the logic to determine the c_i values between the start of the first and the start of the second loop and so it fills more than one cache line; the invocation of the multiplication in the inner loop makes one iteration sufficiently long and constant-time so that inter-probe timings measurably encode the number of iterations of the inner loop.

6.3. Evaluation

To validate that the assumptions are correct, we prototype an end-to-end attack in the SMT-located threat model. The prototype uses two SMT-located processes, attacker and victim. The victim process may be triggered by the attacker to perform ElGamal decryption. We use `Crypto++` to generate a 2048-bit ElGamal instance. The attacker process uses `L1i FLUSH+RELOAD` on the virtual address of the memory-mapped `libgcrypt.so` a shared object that contains code that is executed by the outer loop before the inner loop begins. To collect side channel data, the attacker partitions time into N slots using the `rdtsc` instruction: at the beginning of each slot, the target line is loaded into the cache and the access time is measured; after that, the cache line is flushed and finally, the remaining (if any) time of the slot is spent in a busy loop. As we shall discuss later, we do not have to change any of the settings that might influence the running time of the operation (e.g., c/p states or Turbo Boost). The attacker thus collects N timing samples for load instructions from the target cache line. We establish a threshold for the target system to determine whether the load is likely to have been served from the cache (cache *hit*). We then collect 10,000 measurements by repeatedly triggering the victim. We keep track of each slot across all runs with a set of per-slot counters, all initialized to zero. For each run, whenever the sample of a slot indicates a hit based on the threshold, we increment the value of the corresponding counter.

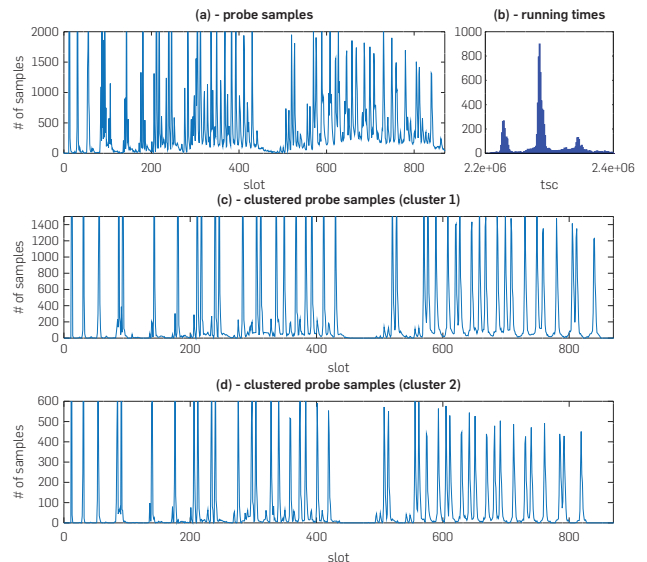
Figure 4a plots the value of the counter of each slot for all runs. While patterns are discernible, the leakage cannot be obtained yet without further data processing. We employ a clustering strategy based on execution time, under the

hypothesis that Figure 4a contains samples that are generated at different clock frequencies. Influencing factors are likely to include p-states, c-states, frequency scaling, and the power state of certain execution units. Figure 4b confirms the hypothesis: it plots a histogram of the running times for the operation. The distribution of running times has a long tail, but most of the mass is concentrated in the three peaks. We use this information to cluster samples and show the results of the clustering in Figures 4c and 4d, showing probes whose running time falls in the interval covered by the highest, and second highest peak, respectively. The peak intervals in Figures 4c and 4d accurately encode the c_i leakage for the key used in the exponentiation. Figures 4c and 4d are identical modulo a scaling factor which depends on the difference in running time. With the leakage thus obtained, we refer back to Section 5 for a detailed description of how the secret exponent is fully recovered.

7. CONCLUSION

We analyze one of the oldest, best understood and most historically used asymmetric encryption schemes, ElGamal.⁹ We reveal that despite its popularity and longevity, when we speak about ElGamal we are referring to several different flavors of it, with key choices being left at the discretion of vendors and implementers. We show how some of these choices create interoperability challenges that lead to insecurity. We propose two “cross-configuration” attacks that are attributable to different, and—from a security standpoint—incompatible, configurations that operate together in the interconnected OpenPGP ecosystem. We believe our work can improve the security of the

Figure 4. FLUSH+RELOAD attack using 10,000 decryption operations with `gcrypt`'s modular exponentiation function. a): number of samples with an icache hit on the target cache line across all runs. b): histogram of the running times (tsc count) for the operation. c) as a) but only considering samples whose total running time lies in the $[2.25 \cdot 10^6, 2.31 \cdot 10^6]$ interval. d) as a) but only considering samples whose total running time lies in the $[2.2 \cdot 10^6, 2.25 \cdot 10^6]$ interval.



OpenPGP community and influence the new revision of the standard that is being drafted at the time of writing.

Acknowledgments

The authors thank Werner Koch, Anil Kurmus, Yutaka Niibe, and Filippo Valsorda for the discussions and feedback, and the CCS'21 reviewers for their comments that helped us improve the paper. 

References

1. OpenPGP server key dump from 15/01/2021. <https://pgp.key-server.io/dump/>, 2021. [Accessed 15/01/2021].
2. Acicmez, O., Koç, Ç., Seifert, J.-P. Predicting secret keys via branch prediction. In *CT-RSA 2007*. M. Abe, ed. Volume 4377 of *LNCS* (2007). Springer, Heidelberg, 225–242.
3. Apecechea, G.I., Inci, M.S., Eisenbarth, T., Sunar, B. Wait a minute! A fast, cross-VM attack on AES. IACR Cryptology ePrint Archive, Report 2014/435 2014, <https://eprint.iacr.org/2014/435>.
4. Bernstein, D.J. Cache-timing attacks on AES. Technical report, 2005.
5. Boneh, D., Joux, A., Nguyen, P.Q. Why textbook ElGamal and RSA encryption are insecure. In *ASIACRYPT 2000*. T. Okamoto, ed. Volume 1976 of *LNCS* (2000). Springer, Heidelberg, 30–43.
6. Callas, J., Donnerhacke, L., Finney, H., Shaw, D., Thayer, R. RFC4880: OpenPGP message format, 2007.
7. De Feo, L., Poettering, B., Sorniotti, A. On the (in)security of ElGamal in OpenPGP. In *ACM CCS 2021*. G. Vigna and E. Shi, eds. ACM, NY, 2021, 2066–2080.
8. De Feo, L., Poettering, B., Sorniotti, A. On the (in)security of ElGamal in OpenPGP. Cryptology ePrint Archive, Report 2021/923, 2021. <https://eprint.iacr.org/2021/923>.
9. ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO'84*. G.R. Blakley and D. Chaum, eds. Volume 196 of *LNCS* (1984). Springer, Heidelberg, 10–18.
10. Genkin, D., Pachmanov, L., Pipman, I., Tromer, E. Stealing keys from PCs using a radio: Cheap electromagnetic attacks on windowed exponentiation. In *CHES 2015*. T. Güneysu and H. Handschuh, eds. Volume 9293 of *LNCS* (2015). Springer, Heidelberg, 207–228.
11. Hachez, G., Quisquater, J.-J. Montgomery exponentiation with no final subtractions: Improved results. In *CHES 2000*. Ç.K. Koç and C. Paar, eds. Volume 1965 of *LNCS* (2000). Springer, Heidelberg, 293–301.
12. Lenstra, H.W. Factoring integers with elliptic curves. *Ann. Math.* 126, (1987), 649–673.
13. Lim, C.H. Lee, P.J. A key recovery attack on discrete log-based schemes using a prime order subgroup. In *CRYPTO'97*. B.S. Kaliski Jr., ed. Volume 1294 of *LNCS* (1997). Springer, Heidelberg, 249–263.
14. Liu, F., Yarom, Y., Ge, Q., Heiser, G., Lee, R.B. Last-level cache side-channel attacks are practical. In *2015 IEEE Symposium on Security and Privacy, SP 2015* (San Jose, CA, USA, May 17–21, 2015), IEEE Computer Society, NY, 2015, 605–622.
15. Messerges, T.S., Dabbish, E.A., Sloan, R.H. Power analysis attacks of modular exponentiation in smartcards. In *CHES'99*. Ç.K. Koç and C. Paar, eds. Volume 1717 of *LNCS* (1999). Springer, Heidelberg, 144–157.
16. Pohlig, S., Hellman, M. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Trans. Inf. Theor.* 24, 1 (Jan. 1978), 106–110.
17. Pollard, J.M. Monte Carlo methods for index computation mod p . *Math. Comput.* 32 (1978), 918–924.
18. Schnorr, C.-P. Efficient identification and signatures for smart cards. In, *CRYPTO'89*. G. Brassard, ed. Volume 435 of *LNCS* (1990). Springer, Heidelberg, 239–252.
19. Shanks, D. Class number, a theory of factorization, and genera. *Proc. Symp. Pure Math.* 20 (1971), 41–440.
20. Tromer, E., Osvik, D.A., Shamir, A. Efficient cache attacks on AES, and countermeasures. *J. Cryptol.* 23, 1 (2010), 37–71.
21. van Oorschot, P.C., Wiener, M.J. On Diffie-Hellman key agreement with short exponents. In *EUROCRYPT'96*. U.M. Maurer, ed. Volume 1070 of *LNCS* (1996). Springer, Heidelberg, 332–343.
22. van Oorschot, P.C. Wiener, M.J. Parallel collision search with cryptanalytic applications. *J. Cryptol.* 12, 1 (Jan. 1999), 1–28.
23. Yarom, Y., Falkner, K. FLUSH+RELOAD: A high resolution, low noise, L3 cache side-channel attack. In *Proceedings of the 23rd USENIX Security Symposium* (San Diego, CA, USA, August 20–22, 2014). K. Fu and J. Jung, eds. (2014). USENIX Association, Berkeley, CA, 719–732.

Luca De Feo, Bertram Poettering, and Alessandro Sorniotti ([feo, poe, aso]@zurich.ibm.com), IBM Research Europe, Zurich, Switzerland.

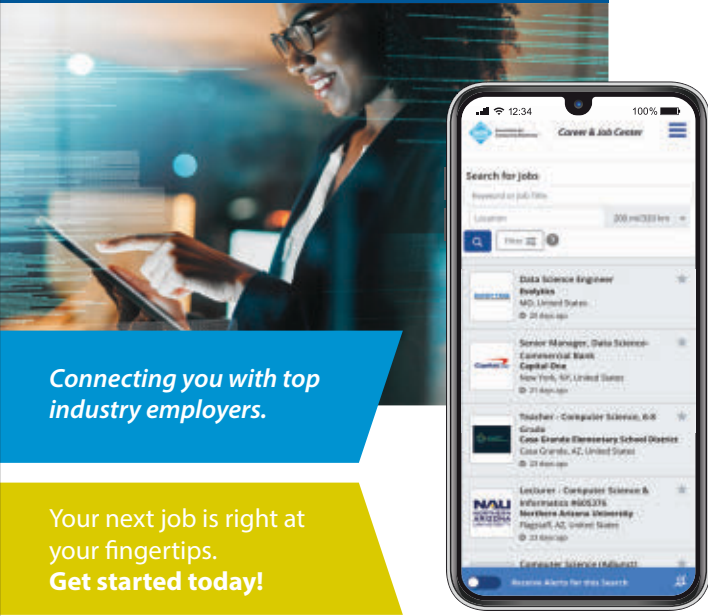
Copyright held by authors/owners. Publication rights licensed to ACM.



Association for Computing Machinery

Career & Job Center

The #1 Career Destination to Find Computing Jobs.



Connecting you with top industry employers.

Your next job is right at your fingertips.
Get started today!

The new ACM Career & Job Center offers job seekers a host of career-enhancing benefits, including:

-  Access to new and exclusive career resources, articles, job searching tips and tools.
-  Gain insights and detailed data on the computing industry, including salary, job outlook, 'day in the life' videos, education, and more with our new Career Insights.
-  Redesigned job search page allows you to view jobs with improved search filtering such as salary, location radius searching and more without ever having to leave the search results.
-  Receive the latest jobs delivered straight to your inbox with **new exclusive Job Flash™ emails**.
-  Get a free resume review from an expert writer listing your strengths, weaknesses, and suggestions to give you the best chance of landing an interview.
-  Receive an alert every time a job becomes available that matches your personal profile, skills, interests, and preferred location(s).

Visit <https://jobs.acm.org/>

Daegu Gyeongbuk Institute of Science & Technology (DGIST) 2023 DGIST Tenure-Track Faculty Public Invitation

It is an honor to have a professor with excellent ability to realize the vision of a Convergence University that changes the world with innovation through convergence education and leading high-tech research along with respect.

1. Positions

- 1) Department of Physics and Chemistry
- 2) Department of Electrical Engineering and Computer Science
- 3) Department of Robotics and Mechatronics Engineering
- 4) Department of Energy Science and Engineering
- 5) Department of Brain Sciences
- 6) Department of New Biology
- 7) Interdisciplinary Engineering

2. Date of Appointment

September 1st, 2023

(Appointment date can be adjusted in consultation with department)

3. Qualification

- ▶ Encourage support for female scientists
- ▶ Ph.D Holder with ability to teach in English required
- ▶ Have no disqualifiers for DGIST full-time faculty appointment

4. Required Documents

- ▶ DGIST application form (Apply after accessing faculty.dgist.ac.kr website)
- ▶ 5 representative achievements
- ▶ (Not Passed to submit less than 5 representative achievements)
- ▶ 3 Letters of Recommendation(LOR)
- ▶ (Please fill out the list of 3 referees or more on your appointment application. Each referee must mail or email a LOR to the department responsible after the applicant moves past the document screening; at the time of filing the application, provide the list of referees only.)

5. How to Apply

- ▶ Apply after accessing <http://dgist.career.co.kr/>
- ▶ Application Period
- ▶ March 31th, 2023 (Fri) ~ April 19th, 2023 (Wed) 18:00(GMT+09:00)

6. Procedure

Document Screening & Department Interview (May~June 2023) ~ Final Interview (Jul., 2023) ~ Faculty Personnel Committee (Jul., 2023)

Above Schedule is variable depending on DGIST internal situation

Results of each step will be individually notified via email

7. Support for Gender Equality & Work-life Balance

- 1) During Pregnancy
- 2) Post Pregnancy
- 3) During Parenting

8. Matters of Consideration

- ▶ If there is no qualified person, no one can be invited

▶ A notice of invitation will be sent to each candidate; an appointment for fixed-term on the agreement to the terms and conditions of employment, with the possibility of further renewal, promotion, or tenure, etc.,based on review.

- ▶ Appointment shall be canceled in case of

1) a false entry or modification is found in the application form,

2) impossible to obtain a Ph.D. as of appointment

▶ If a successful candidate or persons connected with that person involves solicitation, pressure, or the offering of favors in property, etc., that candidate will be subject to cancellation of acceptance and appointment.

- ▶ Appointment will be limited if the candidate is

subject to disqualifications, set out in the State Public Officials Act, Article 33, or DGIST Regulation on Faculty Personnel Management, Article 12.

▶ Other matters not specified in this announcement follow DGIST regulation on Faculty Personnel Management & related regulations

▶ If you have any question, please contact Academic Affairs Team at DGIST

- E-mail : faculty@dgist.ac.kr

- Web-page : <http://dgist.career.co.kr/>

University of Verona - Italy Department of Computer Science Open-rank tenured positions - Associate Professor (PA) or Professor (PO)

We are looking for candidates with an excellent publication record, an internationally visible research profile, a clear potential to promote and lead research activities, and an interest in teaching students at all levels (BS, MS, PhD).

We aim at recruiting a researcher with a strong background in one of the following areas:

NATIONAL ACADEMIES Sciences Engineering Medicine

ARL Distinguished Postdoctoral Fellowships

The Army Research Laboratory (ARL) Distinguished Postdoctoral Fellowships provide opportunities to pursue independent research in ARL laboratories. Fellows benefit by working alongside some of the nation's best scientists and engineers, while enhancing the mission and capabilities of the U.S. Army and the warfighter in times of both peace and war.

Fellows must display extraordinary abilities in scientific research and show clear promise of becoming future leaders. Candidates are expected to have already successfully tackled a major scientific or engineering problem or to have provided a new approach or insight, evidenced by a recognized impact in their field.

Fellowships are one-year appointments, renewable for up to three based on performance. The award includes a \$115,000 annual stipend, health insurance, paid relocation, and a professional travel allowance. Applicants must have completed all requirements for a Ph.D. or Sc.D. degree by October 1, 2023 and may not be more than five years beyond their doctoral degree as of the application deadline. For more information and to apply visit www.nas.edu/arl.

Online applications must be submitted by
July 13, 2023 at 5 PM Eastern Time.



Artificial Intelligence, Theory, Software Engineering and Security (see Research areas).

Verona is a charmingly beautiful, well-connected city of Roman origin, with an exceptionally high quality of life, and a strategic position between central Europe and the Mediterranean.

Applications (and questions regarding the application process) should be sent to segreteria.di@ateneo.univr.it and must contain: A motivation letter, with a specification of the level in which the candidate is interested (Associate Professor or Professor); A curriculum vitae, with a brief research statement; A list of selected publications (at least 5 and no more than 12).

Based on the received expressions of interest, the Department will decide whether to open an official call and which selection procedure to follow, including direct recruitment procedures (e.g., Law 240, art. 7, comma 5bis).

All expressions of interest will be given proper consideration.

The Department of Computer Science of the University of Verona adheres to EU and Italian laws against discrimination based on sex, gender, age, ethnicity, sexual orientation, marital status, disability, or religion, and it appreciates and welcomes diversity and inclusion in all forms.

Woodhaven Lumber & Millwork, Inc. Systems Administrator

Woodhaven Lumber & Millwork, Inc., D.B.A, Woodhaven Lumber & Millwork, is seeking one

(1) full-time Systems Administrator in Lakewood, New Jersey. Full-time employment is a 40-hour work week. The responsibilities and requirements are as follows:

Job Duties:

1. Run day-to-day operations of the Information Technology (IT) Department;
2. Maintain computer hardware, including personal computers, laptops, Chrome boxes, Chromebooks, servers, network, wiring, Sonicwall network appliances, switches, Access Points, scanners, printers, pin pads, GPS units and transceivers, full pump software, Internet Protocol (IP) phone system, fax lines, plain old telephone service (POTS) lines, time clocks, and Truss Manufacturing computers;
3. Maintain computer software, including Microsoft SQL Server, Microsoft Server, Microsoft Active Directory, DNS, DHCP, VPN, Google, Microsoft Office, backup and Antivirus, Kitchen and Truss Designer software, and Epicor Bistrack;
4. Setup and configure new PC workstations
5. Setup new users;
6. Provide help to user's hardware and software requests;
7. Maintain physical inventory cycle count procedure;
8. Create SQL queries for dashboards and Smart Views;
9. Handle network performance, disaster recovery, and security measures;

10. Handle network and domain issues including domain and network security, DHCP, DNS, VPN, NAS, Active Directory;
11. Develop IT Capital Projects for upgrades to IT infrastructure;
12. Work with vendors regarding software and hardware-related purchases and inventory-related technology information sharing; and
13. Work with Microsoft SQL server, Microsoft Windows, Apple OS X, MS Office, Google G-Suite, Microsoft Server, Active Directory Management, MAC and PC hardware, printers, scanners, computer peripherals, and mobile devices (IOS and Android).

This position supervises the work of Computer User Support Specialists.

Education and Experience Requirements: Minimum of a Bachelor degree in Information Science, Information Systems, Information Technology, or Computer Science (the foreign equivalent of a U.S. Bachelor degree accepted), and twelve (12) months experience in Systems Administration, Network Administration, Database Administration, Server Administration, Web Administration, or Network Security Systems Administration.

How to Apply: Email cover letter and resume to Laura Brown, Human Resources Manager, at lbrown@woodhavenlumber.com. Note "Systems Administrator" in the email subject line.

Woodhaven Lumber & Millwork, Inc. is an EEO/AA/ADA/V employer.



ADVERTISING IN CAREER OPPORTUNITIES

How to Submit a Classified Line Ad: Send an e-mail to acmm mediasales@acm.org. Please include text, and indicate the issue/or issues where the ad will appear, and a contact name and number.

Estimates: An insertion order will then be e-mailed back to you. The ad will be typeset according to CACM guidelines. NO PROOFS can be sent. Classified line ads are NOT commissionable.

Deadlines: 20th of the month/2 months prior to issue date. For latest deadline info, please contact: acmm mediasales@acm.org

Career Opportunities Online: Classified and recruitment display ads receive a free duplicate listing on our website at: <http://jobs.acm.org>

Ads are listed for a period of 30 days.

For More Information Contact:
ACM Media Sales
at 212-626-0686 or
acmm mediasales@acm.org



[CONTINUED FROM P. 120] the Alto, and the Ethernet had to fit on a card. So, that was my second takeaway from the trip: radio is a non-starter. But because we had started with radio and then begrudgingly went to cable, that led into the name 'Ethernet.' The original name of Ethernet was actually the Alto-ALOA Network, because it connected Altos and used ALOHA for taking turns. It was only in the so-called invention memo (<http://bit.ly/406xDDi>) that we changed the name to 'Ether.'

I realize you have been telling this story for a long time, but can you summarize what stands out for you now about the technical decisions you made while building Ethernet?

There were three technological decisions that we all thought were important at the time. Now, 50 years later, there are three other things that I see as really important.

Let's start with the original list.

Here were the three things that we built Ethernet around. One of them was coaxial cable. We chose it because there was a device in the cable called the Jerrold tap, thanks to my friend David Liddle, who came from the cable industry. Through the Jerrold tap, you could tap into the coaxial cable without cutting it. That meant we could keep the network

up while we were installing a new PC.

Number two, we chose an encoding technique called Manchester encoding. That's the transceiver—the thing that drives the cable by taking it up to several volts, then letting go. Manchester is the simplest possible encoding. In each bit's cell, the transceiver is on half the time. If it's a one, it's on and then off. If it's a zero, it's off and then on.

It had other features, too. Using Manchester meant that, if you were sending a packet, the network was off half the time—so if you were not transmitting this very instant and someone else was, you could see it. You could do collision detection. The other fea-

"Ethernet brought packets to the desk. Before that, only characters got to the desk, which limited what you could do."

ture of Manchester is that it has an implicit clock in the middle of every bit's cell. Each bit either goes up to down or down to up, and that's the ticking of the clock. That meant that Ethernet did not need a central clock, because every packet had the clock in it during these transitions.

Third, we needed a way for the stations to take turns. That brings us back to randomized transmission. So, randomized transmission, the Jerrold Tap, the Manchester Encoding—that was Ethernet.

And now the list is different.

But it still has three items on it. Three is my favorite number. So, first of all, Ethernet brought packets to the desk. Before that, only characters got to the desk, which limited what you could do. The TELNET Terminal Protocol ran in big time-sharing machines instead of on your little PC. Ethernet changed that.

Two, the packets came really fast. Ethernet began the tradition of abundance. It was 10,000 times faster than what it replaced—not ten, 10,000.

Third, Ethernet joined this novel thing of standardizing. The Ethernet standard went right into the IEEE 802, which did a fabulous job of standardizing it. In hindsight, those three things mattered more than the three technology decisions.



What do you think about when you think about Ethernet's future?

Well, it raises the question, what does the word Ethernet mean? It's been 50 years, so it's had a chance to mean a bunch of different things. Today's Ethernet is not the one that David Boggs—may he rest in peace—and I built in 1973. It's completely different. But some invariants have emerged. One is that Ethernet is an open standard. Another feature is it's faster than anybody needs. It's interoperable. There was a big debate back then about certifying things to be standard. Fine, meet the standard, but you better be interoperable or you're not going to last.

But going back to your question, Ethernet was the transition. The Internet began by connecting time-sharing machines, principally PDP 10s. Then along came the personal computer. The first thing I wrote at Xerox was a TELNET for the Alto—so you could sit at your Alto and log into PDP 10s, as if that were interesting. The personal computer and the Ethernet both happened at exactly the right time. Ethernet was the technology that enabled the transition of the Internet from dumb terminals to personal computers. Now, the word Ethernet is a brand. It's a promise to its users about things like interoperability and speed and standardization.

I'm going to re-ask a question that Robert Colburn asked you nearly 20 years ago (<http://bit.ly/3n7wSet>). What exciting or rewarding or unexpected ways have people found of using the Ethernet, in your view?

Twenty years ago? Time flies. I'll start with automotive Ethernet, which is gratifying because it's a new space. Some of the people who build cars are planning to put Ethernet in them. And I've detected that there's already a fight between different ways of doing it. It's very familiar scene. People recognize the need for standardization, but there's no agreement on

what the standard should be. Which has always been the normal situation of Ethernet.

IoT is also a huge space. And there's a group called the Ethernet Alliance that seems to be doing the right thing. We used to call it co-opetition—you cooperate on the standard and then you compete like hell. That's been typical of the Ethernet. Ethernet was protected by patents, which were put into the public domain. And then my company licensed it, and suddenly we had 100 competitors.

Your next career move takes you back to MIT.

After I retired from the University of Texas, they made me emeritus—which I appreciate, although nobody knows what being 'emeritus' means. Then, I decided my new career would be computational engineer. So I went out and told everybody, and my alma mater, MIT, said, "Why don't you come and do your computational engineering here?" I leapt at the chance. Now I'm a research affiliate of the Computer Science and Artificial Intelligence Laboratory. My first engagement is simulating geothermal wells.

Leah Hoffmann is a technology writer based in Piermont, NY, USA.

© 2023 ACM 0001-0782/23/6 \$15.00

"Today's Ethernet is not the one that David Boggs—may he rest in peace—and I built in 1973. It's completely different, but some invariants have emerged."

Q&A

Still Networking

Bob Metcalfe, who co-invented Ethernet networking technology, continues to take on new challenges as a computational engineer.

ACM A.M. TURING Award recipient Bob Metcalfe—engineer, entrepreneur, and Professor Emeritus at the University of Texas at Austin—is embarking on his sixth career, as a Computational Engineer at the Massachusetts Institute of Technology (MIT) Computer Science and Artificial Intelligence Laboratory (CSAIL). He is always willing to tell the story of his first career, as a researcher at the Xerox Palo Alto Research Center (PARC) where, in 1973, Metcalfe and then-graduate student David Boggs invented Ethernet, a standard for connecting computers over short distances. In the ensuing years, thanks in no small part to Metcalfe's entrepreneurship and advocacy, Ethernet has become the industry standard for local area networks.

Leah Hoffmann spoke to Metcalfe about the development of Ethernet and what it has meant for the future of connectivity.

You published your first paper about Ethernet in *Communications* in July 1976 (<https://bit.ly/403Sxmm>). What do you remember now about the publication process?

Well, I wrote the paper in 1974, but Xerox wouldn't let me publish it, because we had to patent everything first. Then there was a legal review. So the paper benefited because it got rewritten 20 times, and I loved the paper because they let me write it.

You borrowed the name “ether” from 19th-century physics. Then, it was thought to be a passive, omnipresent medium for the propagation of light



and gravity. In your conception, Ethernet would be an omnipresent medium for the propagation of packets of electromagnetic signals. The idea is perfectly suited to a post-Wi-Fi world, but how did you come to it at a time when those

“Because we had started with radio and then begrudgingly went to cable, that led into the name ‘Ethernet’.”

packets were propagated not through the air, but through coaxial cable?

Well, we didn't start with cable; we wanted no wires. And then I went to Hawaii to see the ALOHAnet. I got two things from Hawaii. First was this randomized retransmission technique for taking turns on the channel.

This is the idea—which first reached you via a 1970 National Computer Conference paper by Norm Abramson about ALOHAnet—that if two stations interfere with one another by trying to send at the same time, they should try again at some randomized interval in the future.

That's right. The other thing I saw in Hawaii was an ALOHAnet modem for driving the radio. That modem was bigger than [CONTINUED ON P. 118]

UBICOMP ISWC 2023

CANCUN, MEXICO

www.ubicomp.org
www.iswc.net

8 -12 OCTOBER 2023

UbiComp/ISWC is a premier interdisciplinary venue in which leading international researchers, designers, developers, and practitioners in the field present and discuss novel results in all aspects of ubiquitous, pervasive, and wearable computing. This includes the design, development, and deployment of ubiquitous, pervasive, and wearable computing technologies and the understanding of human experiences and social impacts that these technologies facilitate.

Featuring Full Papers from *IMWUT*, the Proceedings of ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, and *ISWC* Notes and Briefs

Early Registration Ends on July 15th 2023!

Take advantage of our special initiatives on Broadening Participation

Full Conference Rates from \$112.50 for Students and \$200 for ACM Members

Submission Deadlines

Workshops & Symposia

May 26 - June 15, 2023

Student Challenges

June 2, 2023

ISWC Design Exhibition

June 9, 2023

Demos and Posters

June 2, 2023

Doctoral Consortium

June 9, 2023

UbiComp/ISWC
2023



CANCÚN, MÉXICO

Sponsored By



Supported By



Today's Research Driving Tomorrow's Technology

The ACM Digital Library (DL) is the most comprehensive research platform available for computing and information technology and includes the ongoing contributions of the field's most renowned researchers and practitioners.

Each year, roughly 20,000 newly published articles from ACM journals, magazines, technical newsletters and annual conference volumes are added to the DL's complete full text contents of more than 550,000 articles.

The DL also features the fully integrated and comprehensive bibliographic index, *The Guide to Computing Literature*—a continually updated index featuring millions of publication records from over 5,000 publishers worldwide.

For more information, please visit

<https://libraries.acm.org/>

or contact ACM at

dl-info@hq.acm.org

ACM

DL

DIGITAL
LIBRARY