

COMMUNICATIONS

CACM.ACM.ORG

OF THE

ACM

05/2010 VOL.53 NO.05

Beyond Total Capture

A Constructive Critique of Lifelogging

Why Cloud Computing Will Never Be Free

Modeling the Astronomical

Energy-Efficient Algorithms

ACM General Election

Association for
Computing Machinery





ACM SIGCHI Symposium on Engineering Interactive Computing Systems

Berlin, Germany ♦ June 19-23, 2010

General conference chairs

Jean Vanderdonckt (*Université catholique de Louvain, Belgium*) & Noi Sukaviriya (*IBM Research Center, USA*)

Technical program chair

Michael Harrison (*Newcastle University, UK*)

Research papers & practical and experience reports chairs

Jeffrey W. Nichols (*IBM Almaden Research Center, USA*) & Sahin Albayrak (*Technische Universität Berlin, Germany*)

Late breaking results chairs

Angel Puerta (*RedWhale Soft Corporation, USA*) & Gavin Doherty (*Trinity College Dublin, Ireland*)

Demonstrations chairs

Heinrich Arnold (*T-Labs, Germany*), James Lin (*Google, USA*) & Phil Gray (*University of Glasgow, UK*)

Doctoral Consortium chairs

Peter Forbrig (*University of Rostock, Germany*) & Oscar Pastor (*Universidad Politecnica de Valencia, Spain*)

Tutorials chairs

Simone Barbosa (*University PUC-Rio, Brazil*) & Anke Dittmar (*University of Rostock, Germany*)

Workshops chairs

Gaëlle Calvary (*University of Grenoble, France*) & Kevin Schneider (*University of Saskatchewan, Canada*)

Local organisation chairs

Marco Blumendorf & Sahin Albayrak (*DAI Labor Technische Universität Berlin, Germany*)

Publicity chairs

Marco Winckler & Philippe Palanque (*University of Toulouse, France*)

Sponsorship chairs

Kenia Sousa, (*UcL, Belgium*) & Andreas Rieger (*DAI-Labor, Germany*)

Sponsors



Call for participations

EICS is the second international conference devoted to the engineering of usable and effective interactive computing systems. Systems of interest will include traditional workstation-based interactive systems, new and emerging modalities (e.g., gesture), entertaining applications (e.g., mobile and ubiquitous games) and development methods (e.g., extreme programming).

EICS focuses on methods and techniques, and the tools that support them, for designing and developing interactive systems. It brings together people who study or practice the engineering of interactive systems, drawing from the HCI, Software Engineering, Requirements Engineering, CSCW, Ubiquitous / Pervasive Systems and Game Development communities.

EICS encompasses the former **EHCI** (Engineering Human Computer Interaction, sponsored by IFIP 2.7/13.4), **DSV-IS** (International Workshop on the Design, Specification and Verification of Interactive Systems), **CADUI** (International Conference on Computer-Aided Design of User Interfaces) and **TAMODIA** (International Workshop on Task Models and Diagrams) conferences.

The conference will be organized by **DAI-Labor Research Center** and held at **Ernst-Reuter Haus**. The conference will be located at the "Straße des 17. Juni", a famous main axis crossing the heart of the city west to east from Ernst-Reuter-Platz through the Brandenburg Gate up to the palace square (Schlossplatz).

Conference program at a glance

EICS'2010 features only peer-reviewed contributions in the following categories: research papers, demonstrations, doctoral consortium, late breaking results, tutorials, and seven co-located workshops.

Keynote speakers

• User Interface Plasticity: MDE to the limit!

Prof. Joëlle Coutaz, University of Grenoble, France • <http://iihm.imag.fr/coutaz>

• Model Engineering for Model-Driven Engineering

Prof. Axel van Lamsweerde, University of Louvain, Belgium • <http://www.info.ucl.ac.be/~avl>

As part of the social program, there have been scheduled two visits to leading research labs. At the Deutsche Telekom Laboratories and DAI-Labor you will be able to experience ambient assisted living and innovative interaction technologies. At the Human Computer Interaction Lab at Hasso Plattner Institute interactive devices, miniature mobile devices and touch interaction will be presented.

Saturday June 19	Sunday June 20	Monday June 21	Tuesday June 22	Wednesday June 23
Doctoral Consortium	Workshops & Tutorials	Opening Keynote Prof. Joëlle Coutaz Paper session	Paper Session	Keynote Prof. Axel van Lamsweerde Paper Session
Lunch		Lunch & Posters		
Doctoral Consortium	Workshops & Tutorials	Paper Session	Panel Paper Session	Paper Session Closing
Lab tours		Gala dinner		

eics2010.org



Association for
Computing Machinery

Advancing Computing as a Science & Profession

membership application & digital library order form

Priority Code: ACACM10

You can join ACM in several easy ways:

Online
<http://www.acm.org/join>

Phone
+1-800-342-6626 (US & Canada)
+1-212-626-0500 (Global)

Fax
+1-212-944-1318

Or, complete this application and return with payment via postal mail

Special rates for residents of developing countries:

<http://www.acm.org/membership/L2-3/>

Special rates for members of sister societies:

<http://www.acm.org/membership/dues.html>

Please print clearly

Name

Address

City

State/Province

Postal code/Zip

Country

E-mail address

Area code & Daytime phone

Fax

Member number, if applicable

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

I agree with the Purposes of ACM:

Signature

ACM Code of Ethics:

<http://www.acm.org/serving/ethics.html>

choose one membership option:

PROFESSIONAL MEMBERSHIP:

- ☐ ACM Professional Membership: \$99 USD
- ☐ ACM Professional Membership plus the ACM Digital Library: \$198 USD (\$99 dues + \$99 DL)
- ☐ ACM Digital Library: \$99 USD (must be an ACM member)

STUDENT MEMBERSHIP:

- ☐ ACM Student Membership: \$19 USD
- ☐ ACM Student Membership plus the ACM Digital Library: \$42 USD
- ☐ ACM Student Membership PLUS Print CACM Magazine: \$42 USD
- ☐ ACM Student Membership w/Digital Library PLUS Print CACM Magazine: \$62 USD

All new ACM members will receive an
ACM membership card.
For more information, please visit us at www.acm.org

Professional membership dues include \$40 toward a subscription to *Communications of the ACM*. Member dues, subscriptions, and optional contributions are tax-deductible under certain circumstances. Please consult with your tax advisor.

RETURN COMPLETED APPLICATION TO:

Association for Computing Machinery, Inc.
General Post Office
P.O. Box 30777
New York, NY 10087-0777

Questions? E-mail us at acmhelp@acm.org
Or call +1-800-342-6626 to speak to a live representative

Satisfaction Guaranteed!

payment:

Payment must accompany application. If paying by check or money order, make payable to ACM, Inc. in US dollars or foreign currency at current exchange rate.

☐ Visa/MasterCard ☐ American Express ☐ Check/money order

☐ Professional Member Dues (\$99 or \$198) \$ _____

☐ ACM Digital Library (\$99) \$ _____

☐ Student Member Dues (\$19, \$42, or \$62) \$ _____

Total Amount Due \$ _____

Card #

Expiration date

Signature

Departments

- 5 **Editor's Letter**
**Globalization and Offshoring
of Software Revisited**
By Moshe Y. Vardi
-
- 6 **Letters To The Editor**
Roots of Publication Delay
-
- 8 **In the Virtual Extension**
-
- 10 **BLOG@CACM**
**NSF Funding Advice;
21st Century Innovation**
Jeannette M. Wing shares useful
suggestions for department heads.
Daniel Reed discusses the
importance of synergy among
computing specialists and
generalists.
-
- 12 **CACM Online**
Looking for Control
By David Roman
-
- 23 **ACM Election Ballot**
ACM's 2010 General Election
Meet the candidates who
introduce their plans—and stands—
for the Association.
-
- 39 **Calendar**
-
- 118 **Careers**

Last Byte

- 120 **Puzzled**
**Variations on the
Ham Sandwich Theorem**
By Peter Winkler

News



- 13 **Modeling the Astronomical**
Advances in computer technology
have changed the way astronomers
see and study the universe.
By Jeff Kanipe
-
- 16 **Happy Birthday, RDBMS!**
The relational model of data
management, which dates to 1970,
still dominates today and influences
new paradigms as the field evolves.
By Gary Anthes
-
- 18 **Cloud Computing
and Developing Nations**
For a growing number of
organizations worldwide, cloud
computing offers a quick and
affordable way to tap into IT
infrastructure as an Internet service.
But obstacles and challenges remain.
By Samuel Greengard
-
- 21 **Thacker Wins Turing Award**
Microsoft's Charles P. Thacker
named 56th recipient of ACM's
A.M. Turing Award.
By Jack Rosenberger

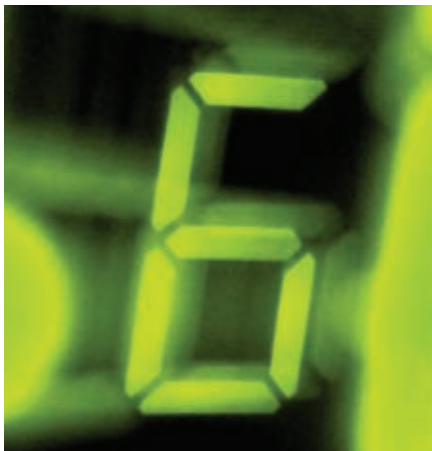
Viewpoints

- 32 **Economic and Business Dimensions**
**Cloud Computing and Electricity:
Beyond the Utility Model**
Assessing the strengths, weaknesses,
and general applicability of the
computing-as-utility business model.
*By Erik Brynjolfsson, Paul Hofmann,
and John Jordan*
-
- 35 **Education**
**How to Make Progress
in Computing Education**
Improving the research base
for computing education
requires securing competitive
funding commitments.
By Cameron Wilson and Mark Guzdial
-
- 38 **Viewpoint**
Can IT Lean Against the Wind?
Lessons from the global
financial crisis.
By Roman Beck
-
- 41 **Viewpoint**
**Is Mobile Email
Addiction Overlooked?**
Studying the prevalence
of mobile email addiction and
the associated possible
implications for organizations.
By Ofir Turel and Alexander Serenko
-
- 45 **Computer Museum Series**
**Great Computing Museums
of the World, Part Two**
The second of a two-part series
highlighting several of the world's
museums dedicated to preserving,
exhibiting, and elucidating
computing history.
By William Aspray



Association for Computing Machinery
Advancing Computing as a Science & Profession

Practice

50 **Enhanced Debugging with Traces**

An essential technique used in emulator development is a useful addition to any programmer's toolbox.
By Peter Phillips

54 **Principles of Robust Timing over the Internet**

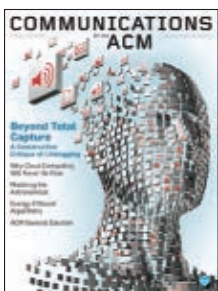
The key to synchronizing clocks over networks is taming delay variability.
By Julien Ridoux and Darryl Veitch

62 **Why Cloud Computing Will Never Be Free**

The competition among cloud providers may drive prices downward, but at what cost?
By Dave Durkee



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

**About the Cover:**

The arduous process of capturing a lifetime of memories is best stored in the human memory, say authors Abigail Sellen and Steve Whittaker in this month's cover story, beginning on p. 70. Brooklyn-based artist Giacomo Marchesi (who also goes by the name James Gray) illustrates the essence of lifelogging, building a metallic image attempting to contain a life in every possible medium.

Contributed Articles

70 **Beyond Total Capture: A Constructive Critique of Lifelogging**

Rather than try to capture everything, system design should focus on the psychological basis of human memory.
By Abigail Sellen and Steve Whittaker

78 **Student and Faculty Attitudes and Beliefs About Computer Science**

The curriculum should inspire students to view CS as both accomplishment and intellectual discipline.
By Clayton Lewis, Michele H. Jackson, and William M. Waite

Review Articles

86 **Energy-Efficient Algorithms**

Algorithmic solutions can help reduce energy consumption in computing environs.
By Susanne Albers

Research Highlights

98 **Technical Perspective**
Learning to Act in Uncertain Environments

By Peter L. Bartlett

99 **Censored Exploration and the Dark Pool Problem**

By Kuzman Ganchev, Yuriy Nevmyvaka, Michael Kearns, and Jennifer Wortman Vaughan

108 **Technical Perspective**
Automated Patching Techniques: The Fix Is In

By Mark Harman

109 **Automatic Program Repair with Evolutionary Computation**

By Westley Weimer, Stephanie Forrest, Claire Le Goues, and ThanhVu Nguyen

Virtual Extension

As with all magazines, page limitations often prevent the publication of articles that might otherwise be included in the print edition. To ensure timely publication, ACM created *Communications'* Virtual Extension (VE).

VE articles undergo the same rigorous review process as those in the print edition and are accepted for publication on their merit. These articles are now available to ACM members in the Digital Library.

The Role of IT in Business Ecosystems

Hyeyoung Kim, Jae-Nam Lee, and Jaemin Han

Monitoring Ambient Air Quality with Carbon Monoxide

Sensor-based Wireless Network
Demin Wang, Dharma P. Agrawal, Wassana Toruksa, Chaichana Chaiwatpongsakorn, Mingming Lu, and Tim C. Keener

Understanding the Dynamics of Information Management Costs

Paul P. Tallon

Roles of the External IT Project Manager

Blaize Reich and Chris Sauer

The Rise of a Health-IT Academic Focus

E. Vance Wilson and Bengisu Tulu

Number of People Required for Usability Evaluation: The 10±2 Rule

Wonil Hwang and Gavriel Salvendy

Is Web-based Supply Chain Integration Right for Your Company?

Charles E. Downing

IT Innovation Persistence: An Oxymoron?

Theophanis C. Stratopoulos and Jee-Haw Lim



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

John White

Deputy Executive Director and COO

Patricia Ryan

Director, Office of Information Systems

Wayne Graves

Director, Office of Financial Services

Russell Harris

Director, Office of Membership

Lillian Israel

Director, Office of SIG Services

Donna Cappel

Director, Office of Publications

Bernard Rous

Director, Office of Group Publishing

Scott Delman

ACM COUNCIL

President

Wendy Hall

Vice-President

Alain Chesnais

Secretary/Treasurer

Barbara Ryder

Past President

Stuart I. Feldman

Chair, SGB Board

Alexander Wolf

Co-Chairs, Publications Board

Ronald Boisvert, Holly Rushmeier

Members-at-Large

Carlo Ghezzi;

Anthony Joseph;

Mathai Joseph;

Kelly Lyons;

Bruce Maggs;

Mary Lou Soffa;

Fei-Yue Wang

SGB Council Representatives

Joseph A. Konstan;

Robert A. Walker;

Jack Davidson

PUBLICATIONS BOARD

Co-Chairs

Ronald F. Boisvert and Holly Rushmeier

Board Members

Jack Davidson; Nikil Dutt; Carol Hutchins;

Ee-Peng Lim; Catherine McGeoch;

M. Tamer Ozsu; Vincent Shen;

Mary Lou Soffa; Ricardo Baeza-Yates

ACM U.S. Public Policy Office

Cameron Wilson, Director

1100 Seventeenth St., NW, Suite 50

Washington, DC 20036 USA

T (202) 659-9711; F (202) 667-1066

Computer Science Teachers Association

Chris Stephenson

Executive Director

2 Penn Plaza, Suite 701

New York, NY 10121-0701 USA

T (800) 401-1799; F (541) 687-1840

Association for Computing Machinery (ACM)

2 Penn Plaza, Suite 701

New York, NY 10121-0701 USA

T (212) 869-7440; F (212) 869-0481

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 GROUP PUBLISHING

Scott E. Delman

publisher@cacm.acm.org

Executive Editor

Diane Crawford

Managing Editor

Thomas E. Lambert

Senior Editor

Andrew Rosenbloom

Senior Editor/News

Jack Rosenberger

Web Editor

David Roman

Editorial Assistant

Zarina Strakhan

Rights and Permissions

Deborah Cotton

Art Director

Andrij Borys

Associate Art Director

Alicia Kubista

Assistant Art Director

Mia Angelica Balaquiot

Production Manager

Lynn D'Addesio

Director of Media Sales

Jennifer Ruzicka

Marketing & Communications Manager

Brian Hebert

Public Relations Coordinator

Virginia Gold

Publications Assistant

Emily Eng

Columnists

Alok Aggarwal; Phillip G. Armour;

Martin Campbell-Kelly;

Michael Cusumano; Peter J. Denning;

Shane Greenstein; Mark Guzdial;

Peter Harsha; Leah Hoffmann;

Mari Sako; Pamela Samuelson;

Gene Spafford; Cameron Wilson

CONTACT POINTS

Copyright permission

permissions@cacm.acm.org

Calendar items

calendar@cacm.acm.org

Change of address

acmcoa@cacm.acm.org

Letters to the Editor

letters@cacm.acm.org

WEB SITE

<http://cacm.acm.org>

AUTHOR GUIDELINES

<http://cacm.acm.org/guidelines>

ADVERTISING

ACM ADVERTISING DEPARTMENT

2 Penn Plaza, Suite 701, New York, NY

10121-0701

T (212) 869-7440

F (212) 869-0481

Director of Media Sales

Jennifer Ruzicka

jen.ruzicka@hq.acm.org

Media Kit acmm mediasales@acm.org

EDITORIAL BOARD

EDITOR-IN-CHIEF

Moshe Y. Vardi

eic@cacm.acm.org

NEWS

Co-chairs

Marc Najork and Prabhakar Raghavan

Board Members

Brian Bershad; Hsiao-Wuen Hon;

Mei Kobayashi; Rajeev Rastogi;

Jeannette Wing

VIEWPOINTS

Co-chairs

Susanne E. Hambrusch; John Leslie King;

J Strother Moore

Board Members

P. Anandan; William Aspray;

Stefan Bechtold; Judith Bishop;

Stuart I. Feldman; Peter Freeman;

Seymour Goodman; Shane Greenstein;

Mark Guzdial; Richard Heeks;

Rachelle Hollander; Richard Ladner;

Susan Landaur; Carlos Jose Pereira de Lucena;

Beng Chin Ooi; Loren Terveen

PRACTICE

Chair

Stephen Bourne

Board Members

Eric Allman; Charles Beeler; David J. Brown;

Bryan Cantrill; Terry Coatta; Mark Compton;

Stuart Feldman; Benjamin Fried;

Pat Hanrahan; Marshall Kirk McKusick;

George Neville-Neil; Theo Schlossnagle;

Jim Waldo

The Practice section of the CACM

Editorial Board also serves as

the Editorial Board of *ACM Queue*.

CONTRIBUTED ARTICLES

Co-chairs

Al Aho and Georg Gottlob

Board Members

Yannis Bakos; Gilles Brassard; Alan Bundy;

Peter Buneman; Ghezzi Carlo;

Andrew Chien; Anja Feldmann;

Blake Ives; James Larus; Igor Markov;

Gail C. Murphy; Shree Nayar; Lionel M. Ni;

Sriram Rajamani; Jennifer Rexford;

Marie-Christine Rousset; Avi Rubin;

Abigail Sellén; Ron Shamir; Marc Snir;

Larry Snyder; Veda Storey;

Manuela Veloso; Michael Vitale;

Wolfgang Wahlster; Andy Chi-Chih Yao;

Willy Zwaenepoel

RESEARCH HIGHLIGHTS

Co-chairs

David A. Patterson and Stuart J. Russell

Board Members

Martin Abadi; Stuart K. Card; Deborah Estrin;

Shafi Goldwasser; Monika Henzinger;

Maurice Herlihy; Norm Jouppi;

Andrew B. Kahng; Gregory Morrisett;

Michael Reiter; Mendel Rosenblum;

Ronitt Rubinfeld; David Salesin;

Lawrence K. Saul; Guy Steele, Jr.;

Gerhard Weikum; Alexander L. Wolf;

Margaret H. Wright

WEB

Co-chairs

Marti Hearst and James Landay

Board Members

Jason I. Hong; Jeff Johnson;

Greg Linden; Wendy E. MacKay



ACM Copyright Notice

Copyright © 2010 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@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 \$100.

ACM Media Advertising Policy

Communications of the ACM and other ACM Media publications accept advertising in both print and electronic formats. All advertising in ACM Media publications is at the discretion of ACM and is intended to provide financial support for the various activities and services for ACM members. Current Advertising Rates can be found by visiting <http://www.acm-media.org> or by contacting ACM Media Sales at (212) 626-0654.

Single Copies

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

COMMUNICATIONS OF THE ACM

(ISSN 0001-0782) is published monthly by ACM Media, 2 Penn Plaza, Suite 701, New York, NY 10121-0701. Periodicals postage paid at New York, NY 10001, and other mailing offices.

POSTMASTER

Please send address changes to *Communications of the ACM* 2 Penn Plaza, Suite 701 New York, NY 10121-0701 USA



Association for
Computing Machinery



Printed in the U.S.A.



DOI:10.1145/1735223.1735224

Moshe Y. Vardi

Globalization and Offshoring of Software Revisited

In 2003–2004, the computing community in the U.S. suddenly realized it had lost its “monopoly” on software. India’s IT industry had been growing at an exponential rate for

a few years by then, driven by offshoring and outsourcing of IT services. Alarming reports by consulting firms predicted the migration of millions of jobs. The political response to the news was quite shrill, referring to “Benedict Arnold CEOs.” Grim jokes circulated about the U.S. offshoring the President’s job. Parents and students became convinced the IT field had no future. U.S. computing enrollments, already hammered by the dot-com and telecomm crashes, took another dive.

In response to these concerns about the viability of computing as a field of study and work in developed countries, ACM Council commissioned in 2004 a Task Force to “look at the facts behind the rapid globalization of IT and the migration of jobs resulting from outsourcing and offshoring.” The Task Force, co-chaired by Frank Mayadas and myself, with the assistance of William Aspray as Editor, was convened in late 2004 and issued its report in February 2006. See <http://www.acm.org/globalizationreport/>

Do the insights produced by the report still ring true four years and a major economic crisis later? To me, the most fundamental insight is that offshoring is just a symptom; the underlying phenomenon is the globalization of computing. Globalization is driven by trade policies, by the evolution of work and business processes, and by IT itself. The dramatic drop in communication costs means that labor can now be viewed as a tradeable good,

just as containerization and air freight reduced shipping costs and increased global trade.

There is no doubt that many people in developed countries lost their jobs to offshoring. Indeed, some categories of jobs, for example, call centers, have migrated more than others. Yet, there is no evidence that offshoring had a measurable impact on aggregate IT employment in developed countries. U.S. employment and wage data showed complete recovery from the dot-com and telecomm crashes by 2005. The report pointed out that once we ignore the many biased and hyped reports and concentrate on concrete data, there is no reason to believe that IT jobs are migrating away.

After the report’s release, many people asked me for a “one-bit answer”: Is offshoring good or bad? Of course, this is not a meaningful question. Good for whom? Undoubtedly, trade contributes to economic growth and greater wealth. Offshoring pulled tens of millions of people out of poverty in India, and that is surely a good thing. But economists admit that “Trade gains may be distributed differentially,” meaning some individuals gain and some lose; some localities gain and some lose. Globalization leads to enhanced competition, and technology leaders risk losing their current dominant position. IT is now a truly global field, business, and industry. Between 1980 and 2000, the market share of the top-five software firms dropped almost 30%.

The bottom line, therefore, is one of a thriving computing field, with projections of continued growth for years to come. The difference is the field is now global, and so is the competition. Such competition is moving up the skill ladder. Companies, including start-ups, are learning how to access and use higher skill levels in developed countries. We are seeing new research labs and increasing national research investment in India and China. We are also seeing an increase in the total worldwide investment in research and a wider distribution of research activities around the world. These are all signs of a thriving global field, with intense global competition.

Can offshoring be ignored? Absolutely not! We must continue monitoring employment and wage trends, as well as global R&D trends, to assess its ongoing impact. To stay competitive in a global IT environment and industry, countries must adopt policies that invest in R&D and foster innovation. Policies that improve a country’s ability to attract, educate, and retain the best IT talent are critical. Education is a primary means for both developed and developing countries to mount a response to offshoring so their work forces can compete globally for IT jobs.

So how do countries and individuals thrive in a global computing world? Invest, innovate, develop, produce, educate, compete. Simple.

Moshe Y. Vardi, EDITOR-IN-CHIEF

Roots of Publication Delay

MOSHE Y. VARDI wrote in his Editor's Letter "Revisiting the Publication Culture in Computing Research" (Mar. 2010) about computer science being a field in which conferences are the primary venue for reporting research. Over the years, the premier conferences have evolved to where their acceptance rates are lower than the corresponding journals and their citation rates are higher. That is why, in the university, the other fields in the tenure councils accept the computer science argument that conference publications deserve the main weight.

When I first got involved in the ACM publication establishment (late 1960s) there was a terrible problem with backlogs in the two ACM research journals of the day: *Communications* and the *Journal of the ACM*. Editorial time—submission to final decision—was running 12–18 months, and publication time—decision to publication—took at least as long. Many loud complaints agreed that 24–36 months for the complete process was bad for such a fast-changing field. However, ACM was almost broke and could not afford more pages, and there was no online venue. Researchers began to look to the SIG conferences as a way to get their material into print sooner, much sooner. By the 1980s, some of these conferences were petitioning the Publication Board to designate them as "refereed" instead of "reviewed."

Studies by Bob Ashenurst and John Rice documented the editorial delays. Even when editors were fastidious about pushing the review cycle along, the authors often took a long time to submit their revisions. Though individual reviews took a long time as well, the authors themselves contributed a lot of delay. The studies also found (with much less supporting data) that authors try different journals and conferences until they find a publisher, and that patient, persistent authors eventually publish over 90% of their papers. Rice calculated an average paper needs four referees, so authors "owe" their colleagues four reviews for every published paper.

A productive author publishing three journal papers a year would thus owe the field 12 reviews a year. Most researchers complain if they have to do half that number. Rice's conclusion, still valid today, was that as long as researchers do not want to do all those (timely) reviews, the editorial phase would not get significantly shorter and is simply the cost of an all-volunteer system.

In 1983 at the start of the *Communications* revitalization plan, we investigated how *Science* magazine gets its submissions reviewed so much quicker. We visited the editors and learned they phoned names from a reviewer database to ask for two-week turnaround. After locating three agreeable reviewers, the editors would FedEx them the manuscript.

We wanted to do likewise, but the ACM executive committee said it was way too expensive. At the time, ACM didn't have enough money to cover even the editors the Council had approved and could not mount a quick review process in *Communications*. Today's online tools make it much easier and less expensive to find reviewers, in which case the bottleneck is again the willingness of reviewers to quickly review and authors to quickly revise.

Peter J. Denning,

ACM Past President, 1980–1982,
Monterey, CA

A fundamental difference between the journal and conference publication systems, as discussed by Moshe Y. Vardi (Mar. 2010), is that conference leadership (chairs and program committees) changes much more frequently than (is practical) for journals. It is difficult (though perhaps not impossible) for a conference to consistently reflect the personal biases and hidden agendas of the same people over a long period of time. The same cannot be said of journals, which have much slower turnover of editors. Such stability allows journals to be, at least in principle, more coherent and less prone to fads, and perhaps less responsive to what the community views as important.

Angelos D. Keromytis, New York

Multi-Departments to Cover the Whole Computational Discipline

Bjarne Stroustrup's Viewpoint "What Should We Teach New Software Developers? Why?" (Jan. 2010) was excellent in its call for curriculum reform but used the wrong model—that a single department is able to fulfill the needs of a mature computing discipline. Other disciplines recognize the need for a multi-departmental model with separate but interrelated departments to support both theory and industrial applications.

Chemistry, physics, and biology departments expand the boundaries of knowledge and create new tools that are then applied by chemical engineering, civil engineering, and clinical medicine departments in industrial settings. Computational (not computer) science departments must be centers of innovation, advancing the general principles of the discipline, while software engineering, IS, and IT departments build on the computational principles from the CS departments to prepare graduates needed by industry.

Undergraduates in chemistry and chemical engineering all take the same general-chemistry courses and labs as freshman, learning general principles with the help of test tubes and Bunsen burners. They then work at the laboratory scale required for research, while chemical-engineering students acquire the skills and knowledge of "bucket" chemistry required for industry.

Professionalism is an important goal, associated, as Stroustrup said, with the application side, not the theoretical side of the discipline. Licensing is for engineers, physicians, and pharmacists in the public domain, not for those working on theory. Coming to a consensus on the appropriate curriculum for different departments makes it easier to develop professional licensure.

A single department is no longer all things to all stakeholders. Needed instead is an ecosystem of interrelated departments supporting the range of theory and applications of the computational discipline.

Mark Segall, Denver, CO

Event Processing for All

In its description of streaming queries, Julian Hyde's article "Data in Flight" (Jan. 2010) included a paragraph on the relationship between streaming queries and complex event processing, saying "CEP has been used within the industry as a blanket term to describe the entire field of streaming query systems. This is regrettable because it has resulted in a religious war between SQL-based and non-SQL-based vendors and, in overly focusing on financial services applications, has caused other application areas to be neglected."

Here, I'd like to offer some perspective on these three assertions:

On the relationship between event processing and streaming SQL. Event processing is a broad term, like data management and signal processing, dealing with computing that performs activities on events. A number of related programming styles are employed in the research community, as well as in commercial products, for implementing event-processing applications, including stream-oriented (based on SQL extensions² and not based on SQL extensions¹), script languages, and rule-based. The Event Processing Technical Society Language Analysis Workgroup gave a tutorial on the languages (<http://www.slideshare.net/opher.etzion/debs2009-event-processing-languages-tutorial>) at the 2009 ACM International Conference on Distributed Event-Based Systems (<http://www.debs.org/2009>).

On vendor competition. I have seen no religious wars over event-processing languages (unlike the classic one between Lisp and Prolog). Vendors compete but generally acknowledge the variety of ways to approach event-processing applications. They also collaborate through the Event Processing Technical society (<http://www.ep-ts.com/>), a consortium including most vendors in the area, as well as its leading academic people, to investigate common functions and steps toward a common modeling language.

On the range of applications. Some have asserted the field is overly focused on financial services to the exclusion all other industries. This might be true for some of the smaller vendors, but the field's major vendors report they

develop applications across a range of industries, including health care, retail, defense, online games, chemical and petroleum, social computing, airline management, and car-fleet management. While applications in the financial services industry were early adopters, the claim of an overly narrow focus for the current state of the practice is simply not correct.

Opher Etzion, Haifa, Israel

References

1. Gedik, B., Andrade, H., Wu, K.-L., Yu, P.S., and Doo, M. SPADE: The system's declarative stream processing engine. In *Proceedings of the SIGMOD Management of Data Conference* (Vancouver, B.C., Canada, June 9–12). ACM Press, New York, 2008, 1123–1134.
2. Jain, N., Mishra, S., Srinivasan, A., Gehrke, J., Widom, J., Balakrishnan, H., Çetintemel, U., Cherniack, M., Tibbetts, R., and Zdonik, S.B. Towards a streaming SQL standard. In *Proceedings of the 34th International Conference on Very Large Databases* (Auckland, New Zealand, Aug. 23–28, 2008), 1379–1390.

Author's Response:

Etzion's point that the Event Processing Technical Society offers a range of approaches to event-processing problems is well taken. However, in framing the problem as "event processing," the Society, as well as the complex event processing vendors, fails to address the needs of the data warehousing/business intelligence community.

Consider how such a BI practitioner might see the world: Business events are embodied as rows in a fact table, and events are captured in log files or through database triggers and transformed through an extract transform load tool or in the target database. SQL is the lingua franca. This perspective is different from the event-processing view, an important distinction because many large, complex problems are expressed in BI terms and because many people have BI skills.

BI technologies alone do not adequately address the needs of this community. Streaming query systems (such as SQLstream) allow them to solve important new problems.

Julian Hyde, San Francisco

Credit Line

The *Communications* cover illustration of Amir Pnueli (Jan. 2010) was inspired by a photograph taken by Dan Porges/Photo Shwartz. 

Communications welcomes your opinion. To submit a Letter to the Editor, please limit your comments to 500 words or less and send to letters@cacm.acm.org.

© 2010 ACM 0001-0782/10/0500 \$10.00



ACM's
interactions
magazine explores
critical relationships
between experiences, people,
and technology, showcasing
emerging innovations and industry
leaders from around the world
across important applications of
design thinking and the broadening
field of the interaction design.
Our readers represent a growing
community of practice that
is of increasing and vital
global importance.

interactions
<http://www.acm.org/subscribe>



In the Virtual Extension

Communications' *Virtual Extension* brings more quality articles to ACM members. These articles are now available in the ACM Digital Library.

The Role of IT in Business Ecosystems

Hyeyoung Kim, Jae-Nam Lee, and Jaemin Han

In a business ecosystem that consists of a large number of loosely interconnected companies, the role of flagship companies is crucial for the survival of the ecosystem. At the ecosystem level, a mechanism for information flow is significant in evaluating a flagship's health, and with it, the ecosystem's competitiveness. IT is used to indicate a flagship's health as it allows for the creation of healthy business ecosystems. This article attempts to provide underlying guidelines in IT that will allow companies to create strong business ecosystems.

Monitoring Ambient Air Quality with Carbon Monoxide Sensor-based Wireless Network

Demin Wang, Dharma P. Agrawal, Wassana Toruksa, Chaichana Chaiwatpongsakorn, Mingming Lu, and Tim C. Keener

Sensors are finding a growing number of applications in many aspects of our daily life. This article explores the design and use of a wireless sensor network in monitoring the presence of the poisonous air pollutant carbon monoxide in the surrounding environment. The authors discuss the different functional units required to implementing such a network and outline calibration, various problems, and associated limitations.

Understanding the Dynamics of Information Management Costs

Paul P. Tallon

Firms continue to experience exponential data growth in all areas of their business. Despite better, faster, and cheaper storage technologies, information management costs continue to rise. In this article, the author considers the challenge of managing information throughout its useful economic life in a way that meets minimum service requirements. Using a tiered-storage infrastructure, firms can use a top-down systems approach to augment service levels for critical information or a bottom-up approach to reduce information management costs for less critical information. Both approaches can also be used to streamline the transfer of information between storage tiers.

Roles of the External IT Project Manager

Blaize Reich and Chris Sauer

Project managers engaged by customers to not only deliver on schedule, budget, and scope targets but also to contribute to value creation will find this article of particular value. Interviews with many senior-level project managers indicate this focus on delivering organizational value—both to the client and within their host organization—has led to three new roles for today's project manager. In addition to the core responsibility of managing an individual project, managers are now expected to take on such roles as account manager, surrogate sponsor, and profession leader. Performing these roles successfully requires new skills such as negotiation, sales, coaching, and strategic networking. The authors examine these new roles and suggest a personal development plan for external project managers.

The Rise of a Health-IT Academic Focus

E. Vance Wilson and Bengisu Tulu

The U.S. health care industry is catching up on its lagging IT investment, and this trend has important ramifications for IT academics and practitioners. Case in point: IT investment in the health care industry has more than doubled in recent years and that investment is paying off. As a result there is a great demand for skilled professionals whose training combines health care domain knowledge with business IT skills. The authors detail examples that reflect the growing promise of (and demand for) academic training in health care topics.

Number of People Required for Usability Evaluation: The 10±2 Rule

Wonil Hwang and Gavriel Salvendy

When a software company wants to evaluate the usability of a new product before launching it into the market, one of the problems the company must solve is determining how many test users or evaluators are required to detect 80% of usability problems in the product. Until now, the '4±1' or 'magic number five' rule has been very popular. However, it cannot be used as a general rule for an optimal number of test users or evaluators because

there is experimental evidence against it. This article tries to answer the question about optimal sample size in usability evaluation and proposes a new '10±2' rule.

Is Web-based Supply Chain Integration Right for Your Company?

Charles E. Downing

The performance of three types of companies is examined in this article: The company with no electronic supply chain integration, one with non-Web-based electronic supply chain integration, and one with Web-based electronic supply chain integration. Results show companies using Web-based integration experience lower cost, higher operational efficiency, a more cooperative partner relationship, and superior overall performance as compared to companies using no electronic integration. Companies using non-Web-based integration exhibit higher customer satisfaction, coordination, cooperation, and commitment with partners, and overall performance as compared to companies using no electronic integration. And finally, companies using non-Web-based integration have a lower volume of complaints and better coordination between partners than companies using Web-based integration.

IT Innovation Persistence: An Oxymoron?

Theophanis C. Stratopoulos and Jee-Haw Lim

Managers' perception of the competitive contribution of IT innovation (ITI) can be likened to a roller coaster ride. Such an observation seems to suggest that companies do not take a systematic approach to ITI because it is easily replicated. However, our analysis shows that ITI is persistent. IT innovative companies are more likely to continue out-innovating their competitors, and this probability is higher during periods when managers' perception regarding the role of IT is pessimistic. Our results show that the probability is low that a non-IT innovative company will become innovative and start out-innovating its competitors within a relatively short period.



THE ACM A. M. TURING AWARD

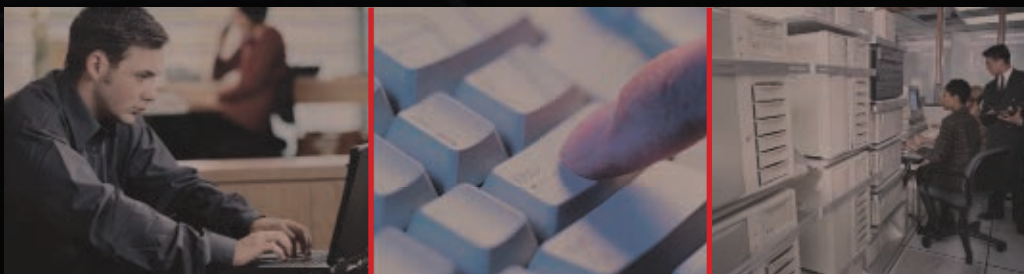
BY THE COMMUNITY...

FROM THE COMMUNITY...

FOR THE COMMUNITY...



ACM, INTEL, AND GOOGLE
CONGRATULATE
CHARLES P. THACKER
FOR THE PIONEERING
DESIGN AND REALIZATION
OF THE FIRST MODERN
PERSONAL COMPUTER—
THE ALTO AT XEROX PARC—
AND SEMINAL INVENTIONS
AND CONTRIBUTIONS TO
LOCAL AREA NETWORKS
(INCLUDING THE ETHERNET),
MULTIPROCESSOR
WORKSTATIONS, SNOOPING
CACHE COHERENCE
PROTOCOLS, AND TABLET
PERSONAL COMPUTERS



“Charles Thacker’s Alto computer embodied the key elements of today’s PCs, and is at the root of one of the world’s most innovative industries. Intel applauds Chuck’s clarity of insight, focus on simplicity, and amazing track-record of designing landmark systems that have accelerated the progress of research and industry for decades.”

Andrew A. Chien
Vice President, Research
Director, Future Technologies Research

For more information see www.intel.com/research.



“Google is pleased to honor contributions that made possible the style of computing that we enjoy today. We are proud to support the Turing Award to encourage continued research in computer science and the related technologies that depend on its continued advancement.”

Alfred Spector
Vice President, Research and
Special Initiatives, Google

For more information, see <http://www.google.com/corporate/index.html> and <http://research.google.com/>.



Financial support for the ACM A. M. Turing Award is provided by Intel Corporation and Google.

The *Communications* Web site, <http://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:10.1145/1735223.1735228

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

NSF Funding Advice; 21st Century Innovation

Jeannette M. Wing shares useful suggestions for department heads. Daniel Reed discusses the importance of synergy among computing specialists and generalists.



**Jeannette M. Wing's
"Twelve Tips for
Department Heads
from an NSF
Perspective"**

[http://cacm.acm.org/blogs/
blog-cacm/54177](http://cacm.acm.org/blogs/blog-cacm/54177)

I was recently asked by the organizers of this summer's CRA Snowbird Workshop for New Department Heads for advice I would give from the National Science Foundation (NSF) perspective. I thought I would share my suggestions with everyone since all academic and industry researchers and practitioners, not just new (and old) department heads, may find them useful. These tips are targeted primarily for academic researchers since they are who NSF primarily funds. In fact, 82% of all federally funded fundamental research in computer science comes from the NSF Computer and Information Science and Engineering (CISE) Directorate. Scary number!

Here's what a department head should advise his or her faculty and what all faculty should do for themselves:

1. Talk to NSF Program Directors. The PDs are the most knowledgeable

about not just the programs they oversee, but also other programs throughout CISE and NSF, and NSF policies and processes. If you have a good idea for a proposal and it's not clear which program is most appropriate for your idea, the PDs can help steer you in the right direction. They can help you navigate through our many rules and procedures.

a. Corollary #1: You may be familiar with one or two "core" programs but as your research changes, you may not realize there are other ongoing "core" or new programs suitable for your new research ideas. Don't think that you need to stick to your current PD or program for funding.

b. Corollary #2: Get to know your Program Director(s). Don't be shy to visit NSF and stop by to meet the PD who is funding you. It's also a good opportunity to meet other PDs not just in CISE but in other directorates and offices related to your interests.

2. Take reporting requirements seriously.

a. Learn to write a good "highlight." It is important for the computing community to be able to explain the nature

of our research to the general public, to the administration, and to Congress. Writing a highlight is a good opportunity to practice this form of communication. Also, it is a real honor if a Principal Investigator's highlight is featured in the President's Budget Request since all highlights submitted to NSF go through many stages of filtering. If your highlight is chosen, add it to your CV.

b. Submit your annual and final reports on time. The funding of a new award will be held up because of missing overdue reports. Alert: Sometime this year, you will be asked to identify a section of your final report that will be published on <http://www.research.gov> so that the public can get a better understanding of research and education supported with taxpayer funds.

3. Learn the basics of the NSF organization. Ditto for other funding agencies. Under the Director of NSF are the Assistant Directors (e.g., one for CISE), then the Division Directors (e.g., three in CISE), and then the PDs.

Here's what a department head should make sure to do:

4. Sign up for the NSF and CISE mailing lists. You can do this via the NSF and CISE Web sites. NSF sends out frequent announcements about funding opportunities and changes to policies and processes. CISE sends out infrequent informal "Dear Colleague Letters" alerting the community to important events and funding opportunities. As a department head, make sure to forward these mailings to your faculty.

5. Mentor junior faculty in writing CAREER and regular NSF proposals.

Give feedback on their research ideas and the ways in which they present them in their proposals. Encourage them to attend the CRA Career Mentoring Workshop run for senior graduate students, post-docs, and assistant professors for advice on writing proposals and other professional development activities.

6. Learn the basics of the federal budget process. Please see my first *Communications* blog entry (<http://cacm.acm.org/blogs/blog-cacm/20487>) on this topic. Make sure to attend any CRA Snowbird session on government affairs. You should learn the vocabulary and what steps happen when.

7. Lead your faculty in building collaborations. Connect the faculty within your department to other departments on campus, to other universities, to industry, to private foundations, and to your local and regional school districts. Computing is transforming every aspect of our lives, so leverage the strengths of your faculty and your institution to build new interdisciplinary partnerships and new research and education collaborations. As a department head, you can spearhead a new large-scale research and/or education effort for your campus, e.g., on the scale of a CISE Expedition, an Engineering Research Center, or an NSF Science and Technology Center.

8. Encourage undergraduates to do research in computing. Make sure your faculty always request supplements to support undergraduate researchers and/or consider submitting proposals to host Research Experiences for Undergraduates Sites.

9. Encourage seniors and first-year graduate students to apply for an NSF Graduate Research Fellowship. Students indicating their intent to pursue graduate studies in computing are underrepresented in this program.

10. Encourage faculty to serve as reviewers for NSF. Serving on a panel can especially be an eye-opening learning experience for junior faculty. Remind your faculty that the PI community and peer review community are one and the same. Service matters.

11. Encourage mid-career and senior faculty to serve as “rotators” to NSF. CISE is always looking for good PDs and Division Directors. The quality of PDs shapes the frontiers of our field. Here is

“We need to consider how we combine our disparate technical skills to attack large-scale problems in a holistic way,” Daniel Reed writes. “This is the beauty and universality of our field.”

the logic: The PD chooses the reviewers and the PD makes the funding recommendations. Clearly good judgment is required for both. In the end what is funded and what is not partly determine where our field moves and what our field looks like.

12. Encourage bold, creative, and visionary thinking. NSF first and foremost looks to fund good people and good ideas.

a. NSF is all about fundamental, long-term research. It is all about high-quality research identified through the merit-review process. Regardless of your perception of NSF, we welcome groundbreaking, high-risk, high-return and potentially transformative ideas.

b. NSF supports not just curiosity-driven, but use-inspired research (“Pasteur’s Quadrant”). Computing research is naturally use-inspired. In tackling societal grand challenges (e.g., energy, environment, health care, and security), we will need both fundamental advances in computing and working with others in other disciplines. Opportunities abound!



**Daniel Reed’s
“Consilience: The Path
to Innovation”**

<http://cacm.acm.org/blogs/blog-cacm/51326>

As a universal intellectual amplifier, computing is the quintessential enabler of 21st century

discovery and innovation. Much of computing’s power accrues from its broad applicability and relevance; its definitions are at least as varied as the talents and interests of its practitioners, in varying proportions spanning mathematics, science, technology, engineering, policy, the humanities and the arts. I would not presume to define computing more precisely for to do so would inevitably omit something important and would unnecessarily constrain its scope.

In this era of hyper-specialization, it is important to remember the combined power of generality and specialization, lest we render true the old joke that education teaches one more and more about less and less until finally one knows everything about nothing and is eligible for a Ph.D. After all, Ph.D. is an abbreviation for the Latin *Philosophiae Doctor* (“Doctor of Philosophy”), and science itself evolved from natural philosophy.

I believe solutions to many of the most challenging problems facing our society—medicine and health care, climate and the environment, economic security and privacy—will require fusing expertise from multiple disciplines, including computing. Filling those disciplinary interstices is both our challenge and our opportunity, and we must embrace diverse intellectual cultures and technical approaches to succeed. We also need both computing specialists and generalists who think systemically.

Cooperation, collaboration, synergy, and consilience—the words are often used and abused, but the sum really is more than just the parts, and the harmonious integration of those parts is often the difference between excellence and mediocrity. It’s why we have building architects, as well as specialists in each technical area. In the same vein, we need to consider how we combine our disparate technical skills to attack large-scale problems in a holistic way. This is the beauty and universality of our field. □

Jeannette M. Wing is a professor at Carnegie Mellon University and the assistant director for Computer and Information Science and Engineering at NSF. **Daniel Reed** is vice president of Technology Strategy & Policy and the eXtreme Computing Group at Microsoft.

© 2010 ACM 0001-0782/10/0500 \$10.00



DOI:10.1145/1735223.1735229

David Roman

Looking for Control

Computer scientists are wrestling to tame the glut of online data, which is overwhelming both individuals and companies (<http://cacm.acm.org/news/46847>). For some users the answer starts with restrictions (<http://calacanis.com/2007/10/03/web-3-0-the-official-definition/>), but that suggestion has been met with predictable, polite opposition grounded in the Web's freewheeling openness (<http://www.demo.com/community/?q=node/9178>).

It's not difficult to find evidence of an overconnected world, and there's a tendency to want to rein it in. More professors are banning distracting laptops from university classrooms (<http://cacm.acm.org/news/79243>). Emergency medical technicians now risk additional life-threatening danger when checking dashboard computers and navigational systems for directions, vital medical information, even baseball scores, while racing patients to the hospital at top speeds. Will "regulators, legislators and safety advocates seek to limit the use of gadgets by most drivers"? (<http://www.nytimes.com/2010/03/11/technology/11distracted.html?pagewanted=1>). Physicians fear losing influence over patients who read online testimonials and incorrect data "perceived as factual" on Web sites not "fully under the doctor's control" (www.physorg.com/news188672089.html). In addition, the Simon Wiesenthal Center recently released its 2010 Digital Terrorism and Hate report, identifying the Web's most dangerous and offensive content (<http://www.foxnews.com/slideshow/scitech/2010/03/15/world-wide-web-hate>).

Restricting access and content is a tempting solution, but restrictions won't end any of this. Connectivity is only the enabler, not the cause. The Internet and mobile communications give some people new ways to misbehave, and examples abound. Legislated penalties don't work (http://recombu.com/news/london-drivers-love-to-chat-legally-or-otherwise_M11236.html). That texting driver in the next lane is not going away soon, law or no law.

Salvation may come from computer science. Researchers are exploring innovations that will make voluminous online data more manageable, for example, by making connections between different data sets (<http://cacm.acm.org/news/80562>). These and other methods will give users new ways to sift through, sort, interrogate, and organize online information.



ACM Member News

ERIC BREWER WINS ACM-INFOSYS FOUNDATION AWARD

Eric Brewer, a professor at the University of California, Berkeley, is the recipient of the 2009 ACM-Infosys Foundation Award in the Computing Sciences for his contributions to the design and development of highly scalable Internet services.

"I spent about a decade working on the use of clusters for scalable, highly available Internet services, starting in 1994 when the Web was new (and didn't need clusters), and ending roughly with the sale of Inktomi [which Brewer co-founded] to Yahoo! in 2003," Brewer said in an email interview. "The surprise, in retrospect, was that it took another five years for the lessons of that decade to become standard practice to the point where companies like Amazon and eBay talked about these issues publicly. Personally, it is the vast use of these ideas that I find most gratifying."

Brewer is currently working to bring information technology to developing nations around the world via his TIER project. "There are a few things I have learned from working in developing countries that I hope to bring back to the U.S.," he said. "Although people talk about the 'digital divide' in terms of nations, the real issue is the divide between urban and rural areas in nearly every nation. Bangalore and Manila, for example, have roughly the same infrastructure, stores, and use of technology as Los Angeles or Amsterdam. But if you go 20 miles outside of Bangalore or Manila you are in a different world altogether—likewise for the middle of California. The rural populations need technology help the most—urban customers will be well addressed by businesses due to their density and ease of reach. I hope that some of the TIER technologies around connectivity and energy will return to the rural U.S."

—Jack Rosenberger

Modeling the Astronomical

Advances in computer technology have changed the way astronomers see and study the universe.

ASTRONOMERS ONCE LEARNED everything they knew about the universe from telescopes, spectroscopes, and other optical instruments. Today, one of the most important items in an astronomer's toolbox is the computer. Galaxy and star formation, supernova explosions, even the origins of the universe—all can be modeled and manipulated to an incredible degree using computers, both powerful mainframes and desktop models. What follows are just a few examples of computer applications that are helping scientists paint bold, new pictures of the cosmos.

Smashing Galaxies

In the early years of the 20th century, astronomers believed that collisions between galaxies were so rare that few, if any, examples might ever be observed. Even in clusters, galaxies are small compared to the vast distances between them, therefore an encounter was deemed unlikely. This mind set quickly shifted, however, once more extensive photographic surveys of the sky revealed that some galaxies appeared to be peculiar amalgams of both spiral-shaped galaxies and spheroidal galaxy-



Gliese 667 C and a pair of stars that were discovered by the High Accuracy Radial Velocity Planet Searcher, the spectrograph for ESO's 3.6-meter telescope at the La Silla Observatory in Chile.

ies. Most of these disturbed specimens could only be explained as products of mergers with other galaxies.

Astronomer and mathematician Alar Toomre, of the Massachusetts Institute of Technology, and his brother Jüri, of the University of Colorado, Boulder, conducted some of the first

computer simulations of galaxy interactions in the early 1970s. They designed numerical programs that could determine the trajectories of a number of test particles, or N -point masses, in which only gravitational interactions are considered. Such programs were run on state-of-the-art mainframes

and required between five and 10 minutes to run a collision scenario with only 350 particles. By the late 1980s, an IBM desktop computer equipped with compiled BASIC could perform the same feat.

The last decade of the 20th century saw significantly more sophisticated computer models that could simulate galaxy interactions over a billion years of time. Initial results of these simulations revealed an unexpected finding. If two spiral galaxies of nearly equal mass passed each other with a low enough velocity, they would essentially fall into each other and, over hundreds of millions of years, merge into a single, massive elliptical galaxy. With this insight, astronomers were not only able to forge an evolutionary link between galaxies of various forms, but also at different epochs in the history of the universe.

The most powerful computers today—supercomputers—have revolutionized the field of galaxy interactions. In one recent example, astronomer Fabio Governato of the University of Washington, Seattle and colleagues used several U.S. supercomputing facilities to solve a long-standing mystery in galaxy formation theory: why most galaxies do not have more stars at their cores. This predicament is most pronounced in a class of diminutive galaxies called dwarf galaxies, the most common type of galaxy in the neighborhood of the Milky Way. Governato's computer simulations showed that supernova explosions in and around the core of a developing dwarf galaxy generate enormous winds that sweep huge amounts of gas away from the center, preventing millions of new stars from forming there.

Governato's project consumed about one million CPU hours, the equivalent of 100 years on a single desktop computer. "This project would have not been possible just a few years ago," he says, "and, of course, completely unthinkable in the 1980s." The enormous growth in computing power, Governato adds, makes the early results obtained by the Toomre brothers using rudimentary computers "even more amazing as their results still hold true."

Searching for Extrasolar Planets

Since 1995, astronomers have discov-

ered more than 400 planets orbiting other stars. Because stars are about a million times brighter than typical planetary bodies, most have been detected using indirect means. One of the most successful of these employs a highly sensitive spectrograph that can detect minute periodic changes in the star's velocity toward and away from Earth—motions caused by changes in the system's center of gravity as the planet orbits the star. These back-and-forth velocity shifts, however, are incredibly slow, amounting to as little as a few meters per second. Still another search method looks for slight wobbles in the star's position in space, a motion also induced by an orbiting body. The extent of the wobble allows astronomers to determine the exoplanet's mass. The positional change is very fine—within 20 milliarcseconds or so, which is about the diameter of a golf ball seen at the distance of the Moon. Such fine-tolerance measurements as these would be impossible without computers.

"The positions of hundreds of lines in spectra are obtained by a truly computer-intensive process of modeling their shapes and noting changes from day to day," says astronomer G. Fritz Benedict of the University of Texas at Austin, who led a team that made the first astrometrically determined mass of an extrasolar planet in 2002.

Computers are also important in helping astronomers understand the nature of exoplanets after they have been discovered. In February 2009, a

The last decade of the 20th century saw significantly more sophisticated computer models that could simulate galaxy interactions over a billion years of time.

team of astronomers associated with a planet-hunting space mission called COROT (for CONvection ROTation and planetary Transits), led by the French space agency CNES, announced that they had detected a planet only 1.7 times Earth's diameter. But the planet, called CoRoT-7b, is hardly Earthlike. It lies only 1.5 million miles from its host star and completes one orbit in a little over 20 hours. Observations indicate the planet is tidally locked, so that it keeps one hemisphere perpetually turned toward its sun. Being so close to its host star, the dayside temperature is some 2600° K, or 4220° F—hot enough to vaporize rocks.

Astronomer Bruce Fegley, Jr. of Washington University and research assistant Laura Schaefer used a computer program called MAGMA to model CoRoT-7b's atmospheric structure. MAGMA was originally developed to study the vaporization of silicates on Mercury's surface, but has since been used, among other things, to model the vaporization of elements during a meteor's fiery passage through Earth's upper atmosphere.

Fegley and Schaefer's results showed that CoRoT-7b's atmosphere is virtually free of volatile compounds, such as water, nitrogen, and carbon dioxide, probably because they were boiled off long ago. But their models indicate plenty of compounds normally found in a terrestrial planet's crust, such as potassium, silicon monoxide, and oxygen. They concluded that the atmosphere of CoRoT-7b has been altered by the vaporization of surface silicates into clouds. Moreover, these clouds may "rain out" silicates, effectively sandblasting the planet's surface.

"Exoplanets are greatly expanding the planetary menagerie," says Fegley. "They provide us with all sorts of odd-ball planets that one could only imagine in the past."

Citizen Science

In 1999, the Space Sciences Laboratory at the University of California, Berkeley, launched a program called SETI@home, which enabled thousands of home computers to sift through unprocessed data collected by radio telescopes in the search for artificial signals from other worlds. The program, which today comprises of more

than five million SETI@home volunteers, essentially constitutes one of the world's largest supercomputers, and is an excellent example of how a network of home computers can be amazingly efficient by spreading around the processing burden.

Another ongoing citizen science program, Galaxy Zoo, uses tens of thousands of home computers, but in this case the processing power lies mostly in the human brain. Originally launched in 2007, Galaxy Zoo is the most massive galaxy classification project ever undertaken. Galaxies come in a mind-boggling variety of shapes and combinations, making classifying them extremely difficult. Galaxy Zoo was tasked with classifying the nearly one million galaxies swept up in the Sloan Digital Sky Survey, the most ambitious survey in the history of astronomy. Computers and the Internet have been instrumental in circulating images of galaxies to thousands of volunteers, but the success of Galaxy Zoo depends on the pattern recognition faculties particular to humans.

"Supercomputers, or even my battered old laptop, can beat large numbers of humans for sheer processing power," says Chris Lintott, an astrophysicist at Oxford University. "The problem we were facing is one of pattern recognition, and for this task, we humans have a huge advantage."

Computerized methods of galaxy classification have been applied to galaxy surveys before, including the Sloan Digital Sky Survey, but their galaxy classification is correct about only 80% of the time, says Lintott. "That may not sound too bad, but the missing 20% are often those that are unusual and thus contain the most information," he says. As Galaxy Zoo team members have discovered, humans are adept at spotting unusual details. For example, volunteers noticed some small, round green objects, now known as "peas," in the background of some Galaxy Zoo images. "They turn out to be the sites of the most efficient star formation in the present-day universe, but it was only because our volunteers spotted them that we knew to study them," says Lintott.

A new incarnation of the project, Galaxy Zoo 2, is now under way and includes a supernova search adjunct. A veritable flood of data is expected. "In

"Exoplanets are greatly expanding the planetary menagerie," says Bruce Fegley, Jr. "They provide us with all sorts of oddball planets that one could only imagine in the past."

the long run, I think we'll need a combination of man and machine," says Lintott. "This sort of development is going to be necessary to cope with the next generation of sky surveys, which should produce data rates of up to 30 terabytes per night." **C**

Further Reading

Governato, F., Brook, C., Mayer, L., Brooks, A., Rhee, G., Wadsley, J., Jonsson, P., Willman, B., Stinson, G., Quinn, T., Madau, P.

Bulgeless dwarf galaxies and dark matter cores from supernova-driven outflows, *Nature* 463, 7278, Jan. 14, 2010.

Benedict, G.F., McArthur, B.E., Forveille, T., Delfosse, X., Nelan, E., Butler, R.P., Spiesman, W., Marcy, G., Goldman, B., Perrier, C., Jefferys, W.H., Mayor, M.

A mass for the extrasolar planet Gl 876b determined from Hubble Space Telescope fine guidance sensor 3 astrometry and high-precision radial velocities, *The Astrophysical Journal* 581, 2, Dec. 20, 2002.

Schaefer, L. and Fegley, B.
Chemistry of silicate atmospheres of evaporating super-Earths, *The Astrophysical Journal* 703, 2, June 2, 2009.

Raddick, M. J., Bracey, G., Gay, P.L., Lintott, C.J., Murray, P., Schawinski, K., Szalay, A.S., Vandenberg, J.
Galaxy Zoo: exploring the motivations of citizen science volunteers, *Astronomy Education Review* 9, 1, Feb. 18, 2010.

Astronomy Education Review Web site
<http://aer.aip.org/aer/>

Based near Boulder, CO, Jeff Kanipe is a science writer and author of *The Cosmic Connection: How Astronomical Events Impact Life on Earth*.

© 2010 ACM 0001-0782/10/0500 \$10.00

Information Retrieval

Data Deluge

The U.S. Blue Ribbon Task Force on Sustainable Digital Preservation and Access released its final report, *Sustainable Economics for a Digital Planet: Ensuring Long-term Access to Digital Information*, a two-year effort that addressed the economic challenges of preserving the world's ever-increasing amount of digital information.

"The data deluge is here," said Fran Berman, vice president for research at Rensselaer Polytechnic Institute and co-chair of the task force. "Ensuring that our most valuable information is available both today and tomorrow is not just a matter of finding sufficient funds. It's about creating a 'data economy' in which those who care, those who will pay, and those who preserve are working in coordination."

A recent study by the International Data Corporation (IDC) found that 3,892,179,868,480,350,000,000 (roughly 3.9 trillion times a trillion) new bits of digital information were created in 2008. IDC predicts that the amount of digital information generated in 2011 will be 10 times the size it was in 2006.

The *Sustainable Economics* report focuses on the economic aspects of digital data preservation, such as how stewards of valuable, digitally based information can pay for preservation over the longer term. The report provides principles and actions to support long-term economic sustainability; context-specific recommendations tailored to specific scenarios analyzed in the report; and an agenda for priority actions and next steps.

The report also concentrates on four different scenarios, each having ever-increasing amounts of preservation-worthy digital assets in which there is a public interest in long-term preservation: scholarly discourse, research data, commercially owned cultural content, such as digital movies and music, and collectively produced Web content, such as blogs.

The report is available at http://brtf.sdsc.edu/biblio/BRTF_Final_Report.pdf.

Happy Birthday, RDBMS!

The relational model of data management, which dates to 1970, still dominates today and influences new paradigms as the field evolves.

FORTY YEARS AGO this June, an article appeared in these pages that would shape the long-term direction of information technology like few other ideas in computer science. The opening sentence of the article, “A Relational Model of Data for Large Shared Data Banks,” summed it up in a way as simple and elegant as the model itself: “Future users of large data banks must be protected from having to know how the data is organized in the machine,” wrote Edgar F. Codd, a researcher at IBM.

And protect them it did. Programmers and users at the time dealt mostly with crude homegrown database systems or commercial products like IBM’s Information Management System (IMS), which was based on a low-level, hierarchical data model. “These databases were very rigid, and they were hard to understand,” recalls Ronald Fagin, a Codd protégé and now a computer scientist at IBM Almaden Research Center. The hierarchical “trees” in IMS were brittle. Adding a single data element, a common occurrence, or even tuning changes, could involve major reprogramming. In addition, the programming language used with IMS was a low-level language akin to an assembler.

But Codd’s relational model stored data by rows and columns in simple tables, which were accessed via a high-level data manipulation language (DML). The model raised the level of abstraction so that users specified *what* they wanted, but not *how* to get it. And when their needs changed, reprogramming was usually unnecessary. It was similar to the transition 10 years earlier from assembler languages to Fortran and COBOL, which also raised the level of abstraction so that programmers no longer had to know and keep track of details like memory addresses.



“People were stunned to learn that complex, page-long [IMS] queries could be done in a few lines of a relational language,” says Raghu Ramakrishnan, chief scientist for audience and cloud computing at Yahoo!

Codd’s model came to dominate a multibillion-dollar database market, but it was hardly an overnight success. The model was just too simple to work, some said. And even if it did work, it would never run as efficiently as a finely tuned IMS program, others said. And although Codd’s relational concepts were simple and elegant, his mathematically rigorous languages, relational calculus and relational algebra, could be intimidating.

In 1969, an ad hoc consortium called CODASYL proposed a hierarchical database model built on the concepts behind IMS. CODASYL claimed that its approach was more flexible than IMS, but it still required programmers to keep track of far more details than the relational model did. It became the basis for a number of commercial products, including the Integrated Database Man-

agement System (IDMS) from the company that would become Cullinet.

Contentious debates raged over the models in the CS community through much of the 1970s, with relational enthusiasts arrayed against CODASYL advocates while IMS users coasted along on waves of legacy software.

As brilliant and elegant as the relational model was, it might have remained confined to computer science curricula if it wasn’t for three projects aimed at real-world implementation of the relational database management system (RDBMS). In the mid-1970s, IBM’s System R project and the University of California at Berkeley’s Ingres project set out to translate the relational concepts into workable, maintainable, and efficient computer code. Support for multiple users, locking, logging, error-recovery, and more were developed.

System R went after the lucrative mainframe market with what would become DB2. In particular, System R produced the Structured Query Language (SQL), which became the de facto standard language for relational databases. Meanwhile Ingres was aimed at UNIX machines and Digital Equipment Corp. (DEC) minicomputers.

Then, in 1979, another watershed paper appeared. “Access Path Selection in a Relational Database Management System,” by IBM System R researcher Patricia Selinger and coauthors, described an algorithm by which a relational system, presented with a user query, could pick the best path to a solution from multiple alternatives. It did that by considering the total cost of the various approaches in terms of CPU time, required disk space, and more.

“Selinger’s paper was really the piece of work that made relational database systems possible,” says David DeWitt, director of Microsoft’s Jim Gray Systems

Laboratory at the University of Wisconsin-Madison. "It was a complete home run." The paper led to her election to the National Academy of Engineering in 1999, won her a slew of awards (including the SIGMOD Edgar F. Codd Innovations Award in 2002), and remains the seminal work on query optimization in relational systems.

Propelled by Selinger's new ideas, System R, Ingres, and their commercial progeny proved that relational systems could provide excellent performance. IBM's DB2 edged out IMS and IDMS on mainframes, while Ingres and its derivatives had the rapidly growing DEC Vax market to themselves. Soon, the database wars were largely over.

Faster Queries

During the 1980s, DeWitt found another way to speed up queries against relational databases. His Gamma Database Machine Project showed it was possible to achieve nearly linear speed-ups by using the multiple CPUs and disks in a cluster of commodity minicomputers. His pioneering ideas about data partitioning and parallel query execution found their way into nearly all commercial parallel database systems.

"If the database community had not switched from CODASYL to relational, the whole parallel database industry would not have been possible," DeWitt says. The declarative, not imperative, programming model of SQL greatly facilitated his work, he says.

The simplicity of the relational model held obvious advantages for users, but it had a more subtle benefit as well, IBM's Fagin says. "For theorists like me, it was much easier to develop theory for it. And we could find ways to make the model perform better, and ways to build functions into the model. The relational model made collaboration between theorists and practitioners much easier."

Indeed, theorists and practitioners worked together to a remarkable degree, and operational techniques and applications flowed from their work. Their collaboration resulted in, for example, the concept of locking, by which simultaneous users were prevented from updating a record simultaneously.

The hegemony of the relational model has not gone without challenge. For example, a rival appeared in the late

1980s in the form of object-oriented databases (OODBs), but they never caught on. There weren't that many applications for which an OODB was the best choice, and it turned out to be easier to add the key features of OODBs to the relational model than to start from scratch with a new paradigm.

More recently, some have suggested that the MapReduce software framework, patented by Google this year, will supplant the relational model for very large distributed data sets. [See "More Debate, Please!" by Moshe Y. Vardi on p. 5 of the January 2010 issue of *Communications*.] Clearly, each approach has its advantages, and the jury is still out.

As RDBMSs continue to evolve, scientists are exploring new roads of inquiry. Fagin's key research right now is the integration of heterogeneous data. "A special case that is still really hard is schema mapping—converting data from one format to another," he says. "It sounds straightforward, but it's very subtle." DeWitt is interested in how researchers will approach the "unsolved problem" of querying geographically distributed databases, especially when the databases are created by different organizations and are almost but not quite alike. And Ramakrishnan of Yahoo! is investigating how to maintain databases in the

cloud, where service vendors could host the databases of many clients. "So 'scale' now becomes not just data volume, it's the number of clients, the variety of applications, the number of locations and so on," he says. "Manageability is one of the key challenges in this space." ■

Further Reading

Codd, E.F.

A relational model of data for large shared data banks. *Comm. of the ACM* 13, 6, June 1970.

Selinger, P.G., Astrahan, M.M., Chamberlin, D.D., Lorie, R.A., Price, T.G.

Access path selection in a relational database management system. *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, 1979.

Ullman, J.D.

Principles of Database and Knowledge-Base Systems: Volume II: The New Technologies, W.H. Freeman & Co., New York, NY, 1990.

Hellerstein, J. M. and Stonebraker, M.

Readings in Database Systems (4th ed.), MIT Press, Cambridge, MA, 2005.

Agrawal, R., et al

The Claremont Report on Database Research, University of California at Berkeley, Sept. 2008. <http://db.cs.berkeley.edu/claremont/claremontreport08.pdf>

Gary Anthes is a technology writer and editor based in Arlington, VA.

© 2010 ACM 0001-0782/10/0500 \$10.00

Looking Ahead With Michael Stonebraker

An adjunct professor at Massachusetts Institute of Technology, Michael Stonebraker is renowned as a database architect and a pioneer in several database technologies, such as Ingres, PostgreSQL, and Mariposa (which he has commercial interests in). As for the database industry's future direction, Stonebraker says one-third of the market will consist of relational database management systems used in large data warehouses, such as corporate repositories of sales information. But the mainstream products in use today, which store table data row by row, will face competition from new, better-performing software that stores it column by column. "You can go wildly faster by rotating your thinking 90 degrees," he says.

Another third of the market he believes will be in online transaction processing, where databases tend to be smaller and transactions simpler. That means databases can be kept in memory and locking can be avoided by processing transactions one at a time. These "lightweight, main memory" systems, Stonebraker says, can run 50 times faster than most online transaction processing systems in use today.

In the final third of the market, there are "a bunch of ideas," depending on the type of application, he says. One is streaming, where large streams of data flow through queries without going to disk. Another type of nonrelational technology will store semistructured data, such as XML and RDF. And a third approach, based on arrays rather than tables, will be best for doing data clustering and complex analyses of very large data sets.

Finally, "if you don't care about performance," says Stonebraker, "there are a bunch of mature, open-source, one-size-fits-all DBMSs."

Cloud Computing and Developing Nations

For a growing number of organizations worldwide, cloud computing offers a quick and affordable way to tap into IT infrastructure as an Internet service. But obstacles and challenges remain.

OVER THE LAST couple of years, cloud computing has taken the business world by storm. The idea of storing and managing data on virtualized servers—often residing on the Internet—isn't particularly new. But thanks to ongoing advances in IT infrastructure and far more sophisticated applications, individuals and organizations around the world now have the ability to connect to data and computing resources *anywhere* and *anytime*.

It's a trend with enormous implications. "Cloud computing provides access to large-scale remote resources in a very efficient and quick manner," explains Karsten Schwan, director of the Center for Experimental Research in Computing Systems at Georgia

Tech University. "It has the potential to dramatically change business models and the way people interact with one another."

Nowhere is this more obvious than in developing nations, where the ability to access resources has often been limited and building out a robust IT infrastructure can be daunting. The emergence of cloud computing changes the stakes for entrepreneurs, small and large businesses, researchers, and governments. "It has the potential to level the playing field because it breaks down barriers to entry," says Steve Bratt, CEO of the non-profit World Wide Web Foundation.

To be sure, cloud computing offers an opportunity to create entirely new types of businesses—and business models—that couldn't have been

imagined or weren't possible only a few years ago. In addition, they open up new markets—including vast numbers of mobile phone users—that previously weren't reachable. However, at the same time, cloud computing presents entirely new challenges and obstacles, particularly in regions coping with limited technical expertise, bandwidth, and IT resources.

A Paradigm Shift?

Although cloud computing has become something of a buzzword over the last couple of years, it's unwise to dismiss it as the latest overhyped tech trend. "It has the potential to create a paradigm shift in the way IT resources are used and distributed," says P.K. Sinha, chief coordinator for research and development at the Center for Development of Advanced Computing at Pune University in India.

Clouds provide a powerful—and often otherwise unattainable—IT infrastructure at a modest cost. In addition, they free individuals and small businesses from worries about quick obsolescence and a lack of flexibility. Yet, at the same time, large organizations can consolidate their IT infrastructure across distributed locations, Sinha points out. Even government entities can benefit by enabling services to consumers on a shared basis. In some cases, cloud-based computing grids enable research that simply wasn't possible in the past.

These days, clouds enable a wide range of services. Already, several industry behemoths—including Amazon, Google, and Microsoft—have introduced cloud-based commercial services. In addition, IBM has established cloud computing centers in Beijing, China; Bangalore, India; Hanoi, Vietnam; Sao Paulo, Brazil; and Seoul, South Korea.



In Kenya, Wilfred Mworia, left, created an iPhone application using only an online iPhone simulator and Steve Mutinda made an Ushahidi application for Java-enabled mobile phones.

For a growing number of organizations worldwide, it's a quick and affordable way to tap into infrastructure as an Internet service. In addition, a growing array of enterprise software vendors, including Salesforce.com and Freshbooks.com, exclusively provide cloud-based services for customers. And Apple and Blackberry make it increasingly simple to store data in the cloud and sync it with multiple devices, including computers and smartphones.

But the story doesn't end there. In India, Africa, and South America, cloud computing allows organizations to connect and collaborate through online applications such as Google Docs. "Many people, including college graduates, do not have access to the latest hardware and software," says Venansius Barya Baryamureeba, dean for the Department of Computing and IT at Makerere University in Kampala, Uganda. What's more, the ability to dial into cloud resources allows organizations lacking an internal IT infrastructure to scale up and compete more effectively.

In fact, the possibilities are limited only by creativity. In the southern Sahara region of Sahel, for example, farmers now use a cloud-based trading system that disseminates information about planting schedules, crop status, harvesting times, and market prices through mobile phones. In India, the Apparel Export Promotion Council has developed a cloud platform that provides computing services and enterprise software to more than 11,000 of its members—most of whom lack the capital and resources to build their own IT infrastructure.

Some cloud initiatives link nations and people in entirely new ways. For example, U.S.-based CrowdFlower has introduced a cloud labor service that connects organizations searching for temporary workers to refugees in Kenya. The iPhone app helps a business outsource a basic task, such as finding phone numbers for marketing departments at various companies and entering them into a spreadsheet. So far, Kenyan workers have completed more than 158,000 unique tasks. These individuals earn as much as U.S. \$28 per week, about eight times

Cloud computing could "level the playing field," says World Wide Web Foundation's Steve Bratt, "because it breaks down barriers to entry."

more than they get from typical jobs in a refugee camp.

Even more remarkable is how clouds connect developers all over the world to technology and markets that would have been entirely out of bounds in the past. *The New York Times* recently reported that Wilfred Mworio, a 22-year-old developer in Nairobi, Kenya, built an iPhone app using only an online iPhone simulator (iPhone service isn't available in Kenya), which he can sell all over the world.

As geography becomes irrelevant, further changes will occur, Baryamureeba notes. In fact, both developed and developing nations will benefit through cloud-based software as a service model. "A software product developed in the USA can be extended and supported by a developer in another place, such as Uganda," he says. "And software can be purchased at a lower cost by eliminating components that are not relevant to a developing country's needs."

Weathering Change

What makes cloud computing so powerful is that it is based on a system of modularity. The use of virtualization and a cloud platform allows organizations to break down services and systems into smaller components, which can function separately or across a widely distributed network. Servers can be located almost anywhere and interfaces can be changed and cus-

Milestones

CS Awards

Mary Jane Irwin and other members of the CS community were recently honored for their research contributions.

ATHENA LECTURER

ACM's Council on Women in Computing has named Mary Jane Irwin, the Evan Pugh Professor of Computer Science of Pennsylvania State University, the 2010–2011 Athena Lecturer for her outstanding research contributions to computer-aided design, computer arithmetic, and computer architecture. The award, which celebrates women researchers who have made fundamental contributions to computer science, includes a \$10,000 honorarium, provided by Google Inc.

NAE 2010 MEMBERS

The National Academy of Engineering has elected 13 new members in the section of computer science and engineering. They are: Andrei Broder, Yahoo!; Irene Greif, IBM; Bill Gropp, University of Illinois at Urbana-Champaign; Laura Haas, IBM; Mike Jordan, University of California at Berkeley; Brewster Kahle, Internet Archive; Tom Mitchell, Carnegie Mellon University; N.R. Narayana Murthy, Infosys; Larry Peterson, Princeton University; Ben Shneiderman, University of Maryland; and Mark Wegman, IBM.

2010 SLOAN RESEARCH FELLOWSHIPS

Sixteen computer scientists were awarded two-year Sloan Research Fellowships. They are: David S. Bindel, Cornell University; David M. Blei, Princeton University; Luis Ceze, University of Washington; Constantinos Daskalakis, MIT; Thomas L. Griffiths, University of California, Berkeley; Eitan Grinspun, Columbia University; Jason I. Hong, Carnegie Mellon University; Karrie Karahalios, University of Illinois at Urbana-Champaign; Jonathan Kelner, MIT; C. Karen Liu, Georgia Tech; Rupak Majumdar, University of California, Los Angeles; Amin Saberi, Stanford University; Ben Taskar, University of Pennsylvania; Brent Waters, University of Texas, Austin; Nikolai Zeldovich, MIT; and Li Zhang, University of Wisconsin, Madison.

tomized on the fly. These services and software are only as far away as an Internet or mobile phone connection.

Nevertheless, challenges and problems remain. One of the biggest, Georgia Tech's Schwan says, is the lack of connectivity and adequate bandwidth in many parts of the world—particularly where dialup networks persist. “Larger data sets require more bandwidth than many locations provide. It's not practical to constantly upload and download all the data. In some instances the technology isn't completely adequate, even in the U.S. and Europe, to enable large data transfers between a company's own infrastructure and the public cloud.”

Another problem is the dependability of local power supplies. In many developing nations, electricity unpredictably switches on and off—and the ability to connect to a remote data center can be dicey. “An organization's IT infrastructure may become inaccessible for some time or data sitting in the cloud could be lost,” says Gautam Shroff, vice president at Tata Consultancy Services Ltd. in Delhi, India. Finally, as with any shared cloud environment, cloud providers and users must address backup, privacy, and security issues.

Nevertheless, cloud computing is poised to take off over the months and

In India, cloud computing is projected to grow from a \$50 million industry in 2009 to a \$15 billion industry by 2013.

years ahead. Baryamureeba believes that clouds—and particularly mobile platforms connecting to them—are a game-changing technology. They extend computing technology into remote areas and provide opportunities for health care, education, and economic development. Meanwhile, “researchers gain access to the same high-performance computing environments as their colleagues in developed nations, and they can be just as competitive and productive,” says Baryamureeba.

Make no mistake, cloud computing is here to stay. In India, for example, cloud computing is projected

to grow from a \$50 million industry in 2009 to a \$15 billion industry by 2013. Other parts of the world are following the same growth trajectory. “It's a catalyst,” observes Bratt of World Wide Web Foundation, “for a wave of innovation and change in developing nations.”

Further Reading

Fingar, P.

Dot.Cloud: The 21st Century Business Platform, Meghan-Kiffer Press, Tampa, FL, 2009.

Morton, G. and Alford, T.

The economics of cloud computing analyzed, October 26, 2009. <http://govcloud.ulitzer.com/node/1147473>

Werth, C.

Number crunching made easy, *Newsweek*, May 2, 2009.

Subramanian, K.

Cloud computing and developing countries—part 1, September 24, 2008. <http://www.cloudave.com/link/Cloud-Computing-and-Developing-Countries—Part-1>

Subramanian, K.

Cloud computing and developing countries—part 2, September 24, 2008. <http://www.cloudave.com/link/Cloud-Computing-and-Developing-Countries—Part-2>

Samuel Greengard is an author and journalist based in West Linn, OR. Nir Kshetri, University of North Carolina at Greensboro, contributed to the development of this article.

© 2010 ACM 0001-0782/10/0500 \$10.00

Obituary

Robin Milner, Turing Award Winner, Dies at 76

Robin Milner, a British computer scientist who won the ACM A.M. Turing Award in 1991, died on March 20th, at the age of 76. Milner's major accomplishments included developing LCF (Logic for Computable Functions), an interactive automated theorem prover, in the early 1970s; creating ML, a general-purpose functional programming language in the late 1970s; and devising the Calculus of Communicating Systems (CCS), a framework for analyzing concurrent systems, in 1980.

Milner's career included being a programmer at Ferranti, a now-defunct British electronics company;

working in the Artificial Intelligence laboratory at Stanford University; leading the department of computer



science at the University of Edinburgh for more than 20 years; and being a professor at Cambridge University and head of its computer laboratory.

His honors included being named a Fellow of ACM and the Royal Society of London, a Foreign member of the Academie Francaise de Sciences, and a Foreign Associate of the National Academy of Engineering.

At the time of his death, Milner was working on bigraphs, a formalism for ubiquitous computing systems. In an interview with Richard Morris, which was published at simple-talk.com last January, Milner discussed his bigraphs research, saying, “I'm working

on theoretical models that aim to give us the ability to engineer and analyze ubiquitous systems more soundly than has often been the case for present-day software. I would dearly love to follow this kind of work through to the front line of design and experience, but I don't think I'll be around for long enough. I'm making up for it by listening and talking to those who will be!”

Communications will present more on the career and accomplishments of Robin Milner in the June issue. At press time, an online memorial for Milner had been established at http://lifestrans.net/robin_milner.

—Jack Rosenberger

Thacker Wins Turing Award

Microsoft's Charles P. Thacker named 56th recipient of ACM's A.M. Turing Award.

AWARDS FOR SCIENTIFIC excellence were recently announced by ACM, SIGACT, the European Association for Theoretical Computer Science, the Computing Research Association, and the Anita Borg Institute for Women in Technology to honor select scientists for their contributions to computer science.

ACM A.M. Turing Award

Charles P. Thacker, a technical fellow at Microsoft's Silicon Valley campus, is the recipient of the 2009 ACM A.M. Turing Award for pioneering work that led to the design and realization of the Alto in 1974, the first modern PC and the prototype for networked personal computers. Thacker is also honored for his contributions to the Ethernet local area network, which he co-invented in 1973; the first multiprocessor workstation; and the prototype for today's most-used tablet PC, with capabilities for direct user interaction.

Thacker created and collaborated on what would become the fundamental building blocks of the PC business. "Charles Thacker's contributions have earned him a reputation as one of the most distinguished computer systems engineers in the history of the field," said ACM President Professor Dame Wendy Hall. "His enduring achievements—from his initial innovations on the PC to his leadership in hardware development of the multiprocessor workstation to his role in developing the tablet PC—have profoundly affected the course of modern computing."

The Turing Award, long recognized as the Nobel Prize in computing, includes a \$250,000 prize, with financial support provided by Intel Corp. and Google Inc.

An in-depth profile and interview with Thacker is scheduled for the July issue of *Communications*.



Turing Award winner Charles P. Thacker

CRA Distinguished Service

The Computing Research Association honored Moshe Y. Vardi, a professor of computer science at Rice University and editor-in-chief of *Communications of the ACM*, with its 2010 Distinguished Service Award. Vardi was nominated for two fundamental contributions to the computing research community. The first was leading the effort to produce a definitive report on offshoring, *Globalization and Offshoring of Software*, which has contributed significantly to debunking myths about the future health of the computing field. His second contribution was leading the effort to redefine *Communications* with the goal of engaging the computing research community to create a compelling magazine for computing.

Theoretical CS Awards

David S. Johnson of AT&T Labs is the recipient of ACM SIGACT's 2009 Knuth Prize for his contributions to theoretic

cal and experimental analysis of algorithms, and Kurt Mehlhorn of the Max Planck Institute is the recipient of the European Association for Theoretical Computer Science's EATCS Award for his distinguished career in theoretical computer science.

"One cannot imagine NP-completeness without David Johnson," Lance Fortnow, a professor of computer science at Northwestern University (and SIGACT chair), told *Communications*. "He developed much of the early theory, helping us deal with NP-complete problems via approximation and experimental algorithms. Johnson promoted theory through SIGACT, the Symposium on Discrete Algorithms, and the DIMACS Implementation Challenges. His legendary reference book on NP-completeness with Michael Garey is an invaluable part of nearly every computer scientist's library."

"Kurt Mehlhorn has done fundamental research in algorithmic theory, including problems from geometry, algebra, graph theory, combinatorial optimization, parallel computing, and VLSI," said Fortnow. "With Stefan Näher, Mehlhorn developed LEDA, the Library of Efficient Data types and Algorithms, providing researchers and businesses with theoretical analyses and implementations of a broad collection of algorithms in geometry, graph theory, and cryptography."

Women of Vision Awards

The Anita Borg Institute for Women in Technology has named Kathleen R. McKeown, Kristina M. Johnson, and Lila Ibrahim as recipients of the 2010 Women of Vision Awards. McKeown, a computer science professor at Columbia University, is the winner in the innovation category. Johnson, the under secretary for the U.S. Department of Energy and the dean of Duke University's Pratt School of Engineering, is the winner in the leadership category. Ibrahim, general manager of Intel's Emerging Markets Platform Group, is the winner in the social impact category. 

Jack Rosenberger is senior editor, news, of *Communications of the ACM*.

© 2010 ACM 0001-0782/10/0500 \$10.00



IHI 2010 – CALL FOR PAPERS

1st ACM International Health Informatics Symposium

November 11-12, 2010

Washington, DC

<http://ihi2010.sighi.org>

General Chair

[Umit V. Catalyurek](#), Ohio State University

Honorary General Chair

[Gang Luo](#), IBM Research

PC Co-Chair

[Henrique Andrade](#), IBM Research

PC Co-Chair

[Neil R. Smalheiser](#), University of Illinois, Chicago

Proceedings Chair

[Tiffany Veinot](#), University of Michigan

Local Arrangements Chair

[Baris E. Suzek](#), Georgetown University

Treasurer

[Alper Yilmaz](#), Ohio State University

ACM HQ Staff Liaison

[April Moskus](#)

Registration Chair

[Bugra Gedik](#), IBM Research

Poster/Demo Chair

[Yuan An](#), Drexel University

Web Chair

[Daby M. Sow](#), IBM Research

Industry Support Chair

[Jinbo Bi](#), Mass General Hospital

Publicity Co-Chair

[Deepak Turaga](#), IBM Research

Publicity Co-Chair

[Albert M. Lai](#), Ohio State University

IMPORTANT DATES

June 2 2010 – Abstract submission

June 4 2010 – Paper/Demo Submission

Aug 6 2010 – Acceptance Notification

IHI 2010 is ACM's premier community forum concerned with the application of computer and information science principles and information and communication technology to problems in healthcare, public health, the delivery of healthcare services and consumer health as well as the related social and ethical issues.

For technical contributions, IHI 2010 is primarily interested in end-to-end applications, systems, and technologies, even if available only in prototype form. Therefore, we strongly encourage authors to submit their original contributions describing their algorithmic and methodological contributions providing an application-oriented context. For social/behavioral scientific contributions, we are interested in empirical studies of health-related information needs, seeking, sharing and use, as well as socio-technical studies of health information technology implementation and use. Topics of interest for this conference cover various aspects of health informatics, including but not limited to the following:

- Accessibility and Web-enabled technologies
- Analytics applied to direct and remote clinical care
- Assistive and adaptive ubiquitous computing technologies
- Biosurveillance
- Brain computer interface
- Cleaning, preprocessing, and ensuring quality and integrity of medical records
- Computational support for patient-centered and evidence-based care
- Consumer health and wellness informatics applications
- Consumer and clinician health information needs, seeking, sharing and use
- Continuous monitoring and streaming technologies
- Data management, privacy, security, and confidentiality
- Display and visualization of medical data
- E-communities and networks for patients and consumers
- E-healthcare infrastructure design
- E-learning for spreading health informatics awareness
- Engineering of medical data
- Health information system framework and enterprise architecture in the developing world
- Human-centered design of health informatics systems
- Information retrieval for health applications
- Information technologies for the management of patient safety and clinical outcomes
- Innovative applications in electronic health records (e.g., ontology or semantic technology, using continuous biomedical signals to trigger alerts)
- Intelligent medical devices and sensors
- Issues involving interoperability and data representation in healthcare delivery
- Keyword and multifaceted search over structured electronic health records
- Knowledge discovery for improving patient-provider communication
- Large-scale longitudinal mining of medical records
- Medical compliance automation for patients and institutions
- Medical recommender system (e.g., medical products, fitness programs)
- Multimodal medical signal analysis
- Natural language processing for biomedical literature, clinical notes, and health consumer texts
- Novel health information systems for chronic disease management
- Optimization models for planning and recommending therapies
- Personalized predictive modeling for clinical management (e.g., trauma, diabetes mellitus, sleep disorders, substance abuse)
- Physiological modeling
- Semantic Web, linked data, ontology, and healthcare
- Sensor networks and systems for pervasive healthcare
- Social studies of health information technologies
- Survival analysis and related methods for estimating hazard functions
- System software for complex clinical studies that involve combinations of clinical, genetic, genomic, imaging, and pathology data
- Systems for cognitive and decision support
- Technologies for capturing and documenting clinical encounter information in electronic systems
- User-interface design issues applied to medical devices and systems

Meet the candidates who introduce their plans—and stands—for the Association.

ACM's 2010 General Election

THE ACM CONSTITUTION provides that the Association hold a general election in the even-numbered years for the positions of President, Vice President, Secretary/Treasurer, and Members-at-Large. On the following pages you will find biographical information and statements from ACM's slate of candidates.

ACM members have the option of casting their vote by postal mail or online. Ballots were mailed to members in early April and contain mail and electronic balloting procedures (please contact acmhelp@electionservicescorp.com if you did not receive a ballot).

The ballot is to be used for the election of ACM Officers (President, Vice President, Secretary/Treasurer) as well as two Members-at-Large.

All ballots must be received by no later than 12:00 noon EDT on May 24, 2010.

The computerized tabulation of the ballots will be validated by the ACM Tellers Committee. Validation by the Tellers Committee will take place at 10:00 a.m. EDT on May 25, 2010.

If you have not done so yet, please take this opportunity to review the candidates and vote via postal mail or the Web for the next slate of ACM officers.

Sincerely,
Gerald Segal, CHAIR, ACM ELECTIONS COMMITTEE

candidates for

PRESIDENT

(7/1/10–6/30/12)

**ALAIN CHESNAÏS**

CTO, SceneCaster
Richmond Hill,
Ontario, Canada

Biography

ACM and SIGGRAPH Activities

- Since July 2008 ACM vice president
- Since July 2005 ACM SIGGRAPH past president
- 2006–2008 ACM secretary/treasurer
- 2005–2006 SIG representative to Council
- 2002–2005 ACM SIGGRAPH president
- 2002–2004 SGB past chair
- 2000–2002 SGB chair
- 2001 SIGGRAPH Conference International Resources Chair
- 1999–2000 SIG Governing Board (SGB) Vice Chair for Operations
- 1999 ACM EC Nominating Committee
- 1998 ACM Executive Search Committee
- 1998 SIGGRAPH Nominating Committee
- 1997 SIGGRAPH Conference International Chair
- 1995–1999 ACM SIGGRAPH Vice Chair
- 1993–1995 Member at Large ACM SIGGRAPH Chapters Committee
- 1993–1995 ACM Director of Professional Chapters
- 1993–1995 ACM Local Activities Board/SIG Board Liaison
- 1992 Organized the local groups' booth for the SIGGRAPH 92 conference
- 1991–1995 Chair of the Paris SIGGRAPH chapter

Professional Experience

- Since June 2007 CTO at SceneCaster
- July 2005–May 2007 Vice President Product Development at Tucows Inc
- March 2004–July 2005 Director of Engineering at ATI
- May 2000–November 2003 Vice President, Engineering at TrueSpectra
- 1996–2000 Director of Engineering, Alias|Wavefront
- 1993–1996 Rendering Manager, Wavefront Technologies
- 1992–1993 CG Consultant
- 1987–1992 CTO, Studio Base 2
- 1983–1987 Research Scientist, Centre Mondial Informatique
- 1982–1985 Research Assistant, CNRS

Education

- 1981 L'École Normale Supérieure de L'Enseignement Technique
- 1980 Diplôme d'Études Approfondies, Université de Paris XI
- 1979 Maîtrise Structure Mathématique de L'Informatique, Université de Paris VII
- 1979 Maîtrise de Mathématiques, Université de Paris VII

Awards/Achievements

- Articles in various journals and conferences.
- Système particulier selected for film show SIGGRAPH 87.
- Opening Sequence selected for film show SIGGRAPH 91.
- Participant in the SIGGRAPH Future Search conference in Snowbird, 1994.
- Participant in the SIGGRAPH strategy meetings in 2000, 2001, 2005 and 2006.

Statement

I was first introduced to ACM through the annual SIGGRAPH conference over 20 years ago. I immediately joined the local SIGGRAPH chapter in Paris, France and started volunteering for various activities. I haven't stopped since. Throughout my volunteer career I have focused on many aspects of ACM activities, ranging from chapters, to SIGs, to Council. I have also been a very active volunteer in SIGGRAPH, our largest SIG. With over 20 years of management experience in the software industry and already 6 years of service on the ACM executive committee, I feel well equipped to tackle the issues that the president of this organization is expected to manage. In my role as SIG Governing Board chair, I led the task force that proposed and implemented a new allocation model for the SIGs, anticipating the technology downturn that was to come when the bubble burst and the impact we expected it to have on ACM finances. That model was adopted by the SIGs and helped the organization weather the storm despite multi-million dollar losses engendered by our largest SIG. I then focused my efforts on ACM SIGGRAPH, after being elected president, and took the SIG's finances from a 2½ million dollar loss when I came in, to generating a modest surplus in my last year.

Today, ACM is a healthy organization that has weathered the recent recession and sees membership rising. The key challenges that I see for ACM in upcoming years have to do with our becoming a truly international organization and attracting younger members into the organization. As a French citizen residing in Canada, I have firsthand experience of what it means to be a non-American member of our organization. I believe that I can use that experience to help ACM truly

become the international organization it should be. Specifically, I believe that we should strengthen our presence in China and India, continuing the effort that we have started during my terms as treasurer, then vice president of the ACM. I also believe that our general presence worldwide can further be enhanced through the changes in the chapters program that I proposed to the executive committee and were then adopted by Council last year.

I also believe that we need to do much more in terms of expanding our online presence to better cater to the needs of younger researchers and practitioners. At SIGGRAPH, I created a Facebook group for SIGGRAPH members to be able to exchange ideas and communicate with each other using the social networking opportunities that it provides. I later proposed that we expand this at the ACM level, both on Facebook where our younger members tend to focus their attention, as well as on LinkedIn where our more confirmed professional members tend to focus. I believe that there is much more that we can do along these lines to further raise the level of awareness of ACM and significantly grow our membership by becoming more relevant to the needs of students as well as young researchers and practitioners.

candidates for

PRESIDENT

(7/1/10–6/30/12)

**JOSEPH A. KONSTAN**

Professor of Computer Science and Engineering
University of Minnesota
Minneapolis, MN, USA

Biography

Education

A.B. (1987) from Harvard College; M.S. (1990) and Ph.D. (1993) from the University of California, Berkeley; all in Computer Science. Dissertation Topic: User Interface Toolkit Architecture (event processing, geometry management, data propagation). Advisor: Larry Rowe.

Employment

McKnight Distinguished University Professor and Associate Head, Department of Computer Science and Engineering, University of Minnesota. Previously, Assistant Professor (1993–1999), Associate Professor (1999–2005), Professor (2005–). Co-Founder and Consulting Scientist, Net Perceptions, Inc. (1996–2000).

Professional Interests

Human-Computer Interaction, Software Tools to Support Online Community, Use of Internet to Promote Public Health (HIV Prevention, Medication Adherence).

Honors/Awards/Achievements

ACM Fellow (2008), ACM Distinguished Scientist (2006), IEEE Distinguished Visitor (2003–2005), ACM Distinguished Lecturer (1997–2006). University awards: Distinguished Professorship, Distinguished Faculty Mentor (for mentoring students from underrepresented minority groups), Taylor Career Development award (contributions to teaching). co-Author of one book, 93 peer-reviewed conference and journal papers. Holds 6 U.S. Patents. Awarded 16 NSF grants (8 as PI), 11 NIH Grants (all as co-Investigator), and 9 other government and industry grants. Created 6 new academic courses. Advised/co-Advised students to 8 Ph.D. degrees and 52 MS degrees.

ACM Activities

ACM Council (2006–2008, 2009–2011). SIG Governing Board: Chair (2006–2008), VC Operations (2004–2006). Membership Services Board: Vice Chair (2004–2006).

SIG Activities

SIGCHI: President (2003–2006), Bulletin Editor (1999–2003), Executive Committee (1999–2009). SIGMultimedia Executive Committee (1999–2007). Conference General Chair: UIST 2003, RecSys 2007. Various program committees, doctoral symposia, etc.

Other Professional/Education Organizations

Federal Demonstration Partnership (partnership of universities and U.S. government agencies focused on reducing administrative burden associated with government grants): U of M faculty representative (2002–2007), alternate (2007–2010); elected Faculty Chair and overall vice-Chair of FDP (2005–2007).

Statement

I am proud to be a member of and a volunteer for ACM, and I am deeply honored by this nomination; I have found my time volunteering for ACM to be immensely rewarding, and I am grateful for the trust of the nominating committee in putting my name forward for ACM's Presidency.

ACM is a healthy and strong society. Our publications and conferences are leading venues for the advancement of computing. Our digital library has become the resource of record for computing literature. And our work on computing education and professional development are key investments into the future of our field.

As a healthy organization, we are not standing still. Recent improvements include not only the new CACM and enhancements to the Professional Development Center, but also less visible but equally important initiatives in China, India, and Europe, efforts to improve our ability to engage and support women in computing, and efforts to speed our responsiveness to new subareas within computing.

More challenges face us—both those brought from outside and those of our own making. As political pressure for professional certification builds, we must continue to explore our position in this debate and our role if such certification comes to pass. We have an obligation both to our members and to the field to oppose “certification” that bears no relationship to actual ability, and to provide meaningful continuing professional education. Economic and environmental realities force us to confront issues about the future of conferences—how can we maintain the essential quality of experience while expanding access?

We have other issues I also hope to take on, supported by both our volunteers and ACM's outstanding professional staff. We must finish the process of reinventing ACM's local activities program, ensuring we understand what local support our members want and need, and providing them with the opportunities to engage with each other and deploy ACM's resources at a local level. We also must continue to improve our presence outside North America. We must build on our efforts in Europe and Asia to ensure that ACM has a friendly, capable, local face to present to its members worldwide. Third, we must do a better job on inreach—being ready to respond to our members, our conference attendees, and our visitors online with useful and timely information.

But most of all, we must continue to do well what we already do well. As the face of computing changes from year to year, the importance of our efforts to support research, professional practice, and education become ever more important. We must build on our strengths and continue to build new strengths.

candidates for

VICE PRESIDENT

(7/1/10–6/30/12)

**NORMAN P. JOUPPI**

Fellow and Director, Exascale Computing Lab
HP Labs
Palo Alto, CA, USA

Biography

Norm received a Ph.D. in Electrical Engineering from Stanford University in 1984, an M.S.E.E. from Northwestern University in 1980, and a B.S.E.E. from Northwestern University in 1979. After graduation he joined DEC's Western Research Lab, and through acquisition Compaq and then Hewlett-Packard. From 1984 through 1996 he was also a Consulting Assistant/Associate Professor in the Department of Electrical Engineering at Stanford University. He currently heads the Exascale Computing Lab at HP Labs.

His current research interests include many aspects of computer system software, hardware, and architecture. He has also led the development of advanced telepresence systems, and has contributed to the architecture and implementation of advanced graphics accelerators. He was the principal architect and lead designer of the DEC MultiTitan and BIPS microprocessors. While at Stanford he was one of the principal architects and lead designer of the Stanford MIPS microprocessor, and developed CAD techniques for VLSI timing verification. He holds 45 U.S. patents and has published over 100 technical papers. Norm is a member of ACM SIGARCH, SIGMICRO, SIGGRAPH, SIGMM, and SIGMETRICS.

ACM Service: ACM SIG Governing Board (SGB) Representative to ACM Council (2007–2009), ACM Council Representative to the Computing Research Association (CRA) Board (2008+), Editorial board of *Communications of the ACM* (2008+), SIG Vice Chair for Operations (2006–2007), ACM SGB Member-at-Large and Conference Advisor (2005–2006), Past Chair of SIGARCH (2007+), Chair of SIGARCH (2003–2007), Vice Chair of SIGARCH (1999–2003), Member of the SIGARCH Board (1993–1999).

Other service: Editorial board member of *IEEE Computer Architecture Letters* (2001+) and *IEEE Micro* (2009+). IEEE TCCA advisory board (2002–2005).

Awards: ACM Fellow. IEEE Fellow. 2005 ISCA Influential Paper award. Compaq 2002 Key Patent award. Two SIGGRAPH/Eurographics Workshop on Graphics Hardware best paper awards.

Statement

Through my previous ACM service I've had the opportunity to learn much about its operation. If elected as Vice President I'd like to place a special emphasis on internationalization, diversity, and enhancing membership value.

A very important issue going forward is international expansion of our membership—business and research contributions to computing are becoming increasingly global. Potential members in developing countries can derive significant benefits from relatively low-cost services such as the ACM Digital Library if ACM membership is priced appropriately. Besides mere membership, I would strive to foster volunteer development and encourage service through the world.

Having two children in high school, I have firsthand experience with the awareness and image of computing among our next generation. It is important for students of all backgrounds to understand and be excited about the contributions of computing to society. To enable this they need first class opportunities to study computer science. If elected, I would work in partnership with existing organizations such as the Computer Science Teachers Association to improve the state of K–12 CS education worldwide and improve the diversity of the CS student body. I would also support efforts to revamp the computer science AP program.

While serving on the ACM Council I've been a strong proponent of adding value to ACM membership through enhanced services while keeping dues low. Over the last several years this strategy has played an important part in growing ACM membership in an era when membership in other professional societies has been waning. If elected as ACM Vice President, I will continue to look for ways to add value to ACM membership (such as the recent successful revitalization of the *Communications of the ACM* and enhancements to the ACM Digital Library) at no marginal cost to our members.

During my career I've worked in both industry and academia. I believe this experience would serve me well as Vice President. As part of this diverse experience I've developed an appreciation of the range of communities served by the ACM, and I believe I can serve them well as Vice President. Finally, I'm prepared to commit the time required to serve at the best of my ability. And I am always eager to listen to your suggestions!

candidates for

VICE PRESIDENT

(7/1/10–6/30/12)

**BARBARA G. RYDER**

J. Byron Maupin Professor of Engineering
 Head, Department of Computer Science
 Virginia Tech
 Blacksburg, VA, USA

Biography

A.B. in Applied Math, Brown University (1969); M.S. in Computer Science (CS), Stanford University (1971); Ph.D. in CS, Rutgers University (1982). Assoc. Member of Prof. Staff at AT&T Bell Labs, Murray Hill (1971–1976). Asst. Prof. (1982–1988), Assoc. Prof. (1988–1994), Prof. (1994–2001), Prof. II (2001–2008), Rutgers University, Head - Dept. of CS, Virginia Tech (2008–), <http://people.cs.vt.edu/~ryder/>
 Fellow of the ACM (1998) for seminal contributions to interprocedural compile-time analyses. Member, Board of Directors, Computer Research Assn (1998–2001). ACM Presidential Award (2008). SIGPLAN Distinguished Service Award (2001). Rutgers Graduate Teaching Award (2007). Rutgers Leader in Diversity Award (2006). Professor of the Year Award from CS Grad Students (2003).

ACM Secretary/Treasurer (2008–2010), Council Member-at-Large (2000–2008). Chair, Federated Computing Research Conf (FCRC 2003). SIGPLAN Chair (1995–1997), Vice Chair for Confs (1993–1995), Exec Comm (1989–1999). General Chair of: SIGSOFT Int'l Symp on Software Testing and Analysis (ISSTA, 2008), SIGPLAN Conf on History of Programming Languages III (HOPL-III, 2007), SIGPLAN Conf on Programming Language Design and Implementation (PLDI, 1999, 1994). Program Chair of: HOPL-III (2007), PLDI (1991). Member, Outstanding Contribution to ACM Award Comm and ACM-W Athena Award Comm. ACM National Lecturer (1985–1988).

Member, Ed Bd of Science of Programming (2009–), *ACM Trans on Programming Languages and Systems* (TOPLAS, 2001–2007), and *IEEE Trans on Software Engineering* (2003–2008).

Selected panelist: CRA Workshops on Academic Careers for Women in CS (1993, 1994, 1996, 1999, 2003), SIGSOFT New SE Faculty Symp (2003, 2005, 2008). Chair, Virginia Tech College of Engineering High Performance Computing Comm (2009–). Member, ADVANCE VT Faculty Advisory Comm (2008–). Member: SIGPLAN, SIGSOFT, SIGCSE, ACM, IEEE Computer Society, American Women in Science, EAPLS.

Statement

As ACM Vice President, I will work closely with the President and provide leadership for Executive Committee special projects (e.g., the Membership Task Force). I will continue to support good communication among the Executive Committee, Council, SIG leadership, members and staff. My extensive experience as a SIG leader, General Chair of FCRC 2003, 8 years as ACM Council member, and 2 years as Secretary/Treasurer have prepared me well for these tasks.

I am determined to maintain ACM as the leading computing society, and our representative on issues of public policy worldwide. There are several challenges: expanding ACM into a truly international organization, strongly supporting computing education (K–12, college, postgraduate), providing better services to our practitioner members, and supporting the SIGs.

Recent efforts by ACM that have established relationships with professionals in India and China must be strengthened and supported by recruiting new members from these regions into SIG/ACM leadership. Additionally, we should hold ACM conferences outside of North America, co-sponsored with local professional societies. New contacts for international members should be sought in areas such as Eastern Europe/Russia, South Asia/Australia and South America.

ACM activities on education should emphasize the importance of our discipline in the 21st century. We need to continue ACM efforts on computing curricula and building bridges to K–12 educators (i.e., Computer Science Teachers Association). New emphases should include: supporting CSTA-like organizations internationally and encouraging local/regional technical conferences for undergraduate and graduate students (e.g., www.research.ibm.com/masplas).

ACM has revitalized the Local Chapters program, offered new opportunities to mentor younger professionals through MemberNet, enhanced the Digital Library and Portal, and redesigned *CACM* to meet the needs of all members better. Such efforts must continue.

The SIGs must remain a strong, integral part of ACM: developing volunteer leaders, providing valuable research content and tools for the Digital Library, and recruiting students to ACM membership. My 10 years of SIG leadership and 35 years of active ACM membership attest to my commitment to SIG concerns.

I ask for your vote to work for all of these goals.

candidates for

SECRETARY/TREASURER

(7/1/10–6/30/12)

**CARLO GHEZZI**

Chair of Software Engineering
Department of Electronics and Information
Politecnico di Milano
Milano, Italy

Biography

Carlo Ghezzi is a Professor and Chair of Software Engineering at Politecnico di Milano, Italy. He is the Rector's Delegate for research, past member of the Academic Senate and of the Board of Governors of Politecnico. He has also been Department Chair and Head of the Ph.D. Program. He held positions at University of California Los Angeles, University of North Carolina at Chapel Hill, University of Padova, ES-LAI-Buenos Aires, University of California Santa Barbara, Technical University of Vienna, University of Klagenfurt, University of Lugano.

Ghezzi is an ACM Fellow, an IEEE Fellow, and a member of the Italian Academy of Sciences. He was awarded the ACM SIGSOFT Distinguished Service Award. He has been a member of the ACM Nominating Committee and of the ACM Software Systems Award Committee. He has been on the evaluation board of several international research projects and institutions in Europe, Japan, and the USA.

Ghezzi is a regular member of the program committee of important conferences in the software engineering field, such as the International Conference on Software Engineering and Foundations of Software Engineering/ European Software Engineering Conference, for which he also served as Program and General Chair.

He served on numerous other program committees and gave several keynote presentations.

Ghezzi has been the Editor-in-Chief of the *ACM Transactions on Software Engineering and Methodology* (from 2001 till 2006) and is an Associate Editor of *Communications of the ACM*, *IEEE Transactions on Software Engineering*, *Science of Computer Programming*, *Service Oriented Computing and Applications*, and *Software Process Improvement and Practice*.

Ghezzi's research has been focusing on software engineering and programming languages. Currently, he is active in evolvable and distributed software architectures for ubiquitous and pervasive computing. He co-authored over 160 papers and 8 books. He coordinated several national and international (EU funded) research projects. He has recently been awarded a prestigious Advanced Grant from the European Research Council.

Statement

ACM is the leading professional society in the field of computing. For over 60 years, it has been serving the scientific community: researchers, educators, engineers, and professional developers. ACM conferences, workshops, journals, magazines, newsletters, and digital library played a fundamental role in accumulating and disseminating knowledge, creating new knowledge, and linking people. ACM will continue to preserve this fundamental body of knowledge, and will also assist us in future advances in our field.

ACM is increasingly becoming a worldwide professional society. It will need in the future to become even more open, offering services and opportunities everywhere in the world. By its very nature, computing is at the heart of technology that connects the world. It should by no means become the source of divisions and discriminations. ACM should elaborate policies to support worldwide knowledge sharing and scientific cooperation, breaking all barriers of race, culture, economy, gender, and age. It should aggressively involve new communities, understand their needs, and be flexible in adapting its global strategies to the local cultures.

Higher-level education in computing has been experiencing difficulties in many regions of the world, because it is not attracting enough brilliant young people, and especially women. Computing is often viewed as lacking deep challenging underpinnings. The society at large often has misconceptions about our profession and its roots in science. ACM should take the lead of a pride initiative at all levels, finding new ways to communicate with the society, and especially with the new generations.

candidates for

SECRETARY/TREASURER

(7/1/10–6/30/12)

**ALEXANDER L. WOLF**

Professor
Department of Computing
Imperial College London
United Kingdom

Biography

Alexander Wolf holds a Chair in Computing at Imperial College London, U.K. (2006–present). Prior to that he was a Professor at the Univ. of Lugano, Switzerland (2004–2006), Professor and C.V. Schelke Endowed Chair of Engineering at the Univ. of Colorado at Boulder, U.S. (1992–2006), and Member of the Technical Staff at AT&T Bell Labs in Murray Hill, NJ, U.S. (1988–1992).

Wolf earned his MS (1982) and Ph.D. (1985) degrees from the Univ. of Massachusetts at Amherst, U.S., from which he was presented the Dept. of Computer Science Outstanding Achievement in Research Alumni Award (2010).

With colleagues and students, Wolf works in several areas of experimental and theoretical computer science, including software engineering, distributed systems, networking, and databases (see <http://www.doc.ic.ac.uk/~alw/> for links to his papers). He is best known for seminal contributions to software architecture, software deployment, automated process discovery (the seed of the business intelligence field), distributed publish/subscribe communication, and content-based networking. He has recently begun to investigate fault localization in MANETs and large-scale experiment automation.

Wolf serves as Chair of the ACM SIG Governing Board (SGB) and Chair of the ACM Software Systems Award Committee. He is a member of the ACM Council and ACM Executive Committee. He is also a member of the newly formed ACM Europe Council. Wolf serves on the editorial boards of the Research Highlights section of *CACM* and of the IEEE journal *TSE*. Previously, he served as Vice Chair and then Chair of ACM SIGSOFT, a member of the SGB Executive Committee, an SGB representative on the ACM Council, member of the editorial board of ACM TOSEM, and Chair of the ACM TOSEM EiC Search Committee. He has chaired and been a member of numerous international program committees.

Wolf is a Fellow of the ACM, Chartered Fellow of the British Computer Society, holder of a U.K. Royal Society–Wolfson Research Merit Award, winner of an ACM SIGSOFT Research Impact Award, and is an ACM Distinguished Speaker.

Statement

As the public has begun to recognize computing's central role in supporting and advancing society, we have been reshaping ACM as a key player in supporting and advancing the computing discipline.

In my recent leadership roles I have contributed to these efforts, including formation of regional councils (so far, Europe and India), reconceptualization of regional and student chapters, an initiative to proactively nurture conferences and SIGs in new areas of computing, sponsorship of the new Computer Science Teachers Association, enrichment of the DL, and a revamp of *CACM* to be more relevant, timely, informative, accessible, and authoritative.

Essential to these and future initiatives is a clear and convincing long-term strategy that draws together the talents of volunteers and staff, supported by sufficient financial resources.

The world financial crisis has been a necessary focus of my term as SGB chair and ACM Council member. Managing financial risk while maintaining the integrity and momentum of the community is a difficult challenge, but one that we have handled extremely well through the joint efforts of the volunteer leaders and headquarters staff.

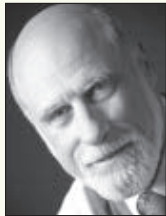
We must continue to be sensitive to how the crisis will affect different sectors and regions of our community, and the impact on conference planning and regional growth initiatives. Our efforts at securing a significant fund balance in the past have contributed to the fundamental stability of the organization today; we are fortunate to be in a position to absorb much of the pain of the crisis, yet continue to grow and improve the organization's services.

I have been an ACM volunteer for much of my career. The opportunity to continue contributing by serving as ACM Secretary/Treasurer would be a privilege and an honor.

candidates for

MEMBERS AT LARGE

(7/1/10–6/30/14)

**VINTON G. CERF**

Vice President and
Chief Internet Evangelist
Google
Reston, VA, USA

Biography

Vinton G. Cerf has served as VP and chief Internet evangelist for Google since October 2005. He is responsible for identifying new enabling technologies to support the development of advanced, Internet-based products and services from Google. In the Internet world, he is an active public face for Google. Cerf is the former senior VP of Technology Strategy for MCI and VP of MCI Digital Information Services. Recognized as one of the “Fathers of the Internet,” Cerf is the co-designer of the TCP/IP protocols and the Internet architecture. During his tenure from 1976–1982 with the U.S. Department of Defense’s Advanced Research Projects Agency (DARPA), Cerf played a key role leading the development of Internet and Internet-related packet data and security technologies. In December 1997, President Clinton presented the U.S. National Medal of Technology to Cerf and his colleague, Robert E. Kahn, for founding and developing the Internet. Kahn and Cerf were named the recipients of the ACM A. M. Turing Award in 2004 for their work on the Internet protocols. In November 2005, President Bush awarded Cerf and Kahn the Presidential Medal of Freedom—the highest civilian award given by the U.S. to its citizens. In April 2008, Cerf and Kahn received the prestigious Japan Prize.

ACM also awarded Cerf’s work the ACM Software System Award and ACM SIGCOMM Award. Cerf served as chairman of the board of the Internet Corporation for Assigned Names and Numbers (ICANN) from 2000–2007 and was founding president of the Internet Society from 1992–1995, serving a term as chairman of the Board in 1999. Cerf holds a Bachelor of Science degree in Mathematics from Stanford University and Master of Science and Ph.D. degrees in Computer Science from UCLA and is the recipient of over a dozen honorary degrees.

Statement

I have been a member of ACM since 1967 served as a member of Council in the distant past during which time my primary policy objective was to create the Fellow grade of ACM membership. I also served for four years as chairman of ACM SIGCOMM.

As Member-at-Large, I consider that my primary function will be to convey to Council and ACM leadership the policy views of the general membership. To this end, I will invite open dialog with any and all members of ACM so as to be informed of their views. I offer to do my best to represent these views and also to exercise my best judgment in the setting of ACM policy and assisting the staff in their operational roles.

It seems clear that ACM can and must take increasing advantage of the online environment created by the Internet, World Wide Web, and other applications supported by this global networking infrastructure. This suggests to me that Council and ACM leadership should be looking for ways to assist Chapters and Special Interest Groups to serve as conduits for two-way flows of information, education, training, and expertise. The power and value of ACM membership flows from the contributions and actions of its members.

As a consumer of ACM publications, I am interested in increasing the accessibility and utility of ACM’s online offerings, including options for reducing the cost of access to content in this form. By the same token, I am interested in making the face-to-face events sponsored by ACM and its affiliates useful during and after the events have taken place. The possibility of audio, video, and text transcripts of presentations (perhaps starting with keynotes) seems attractive. If these forms of content can be made searchable, their value may well increase and persist over longer periods of time.

**SATOSHI MATSUOKA**

Professor of Mathematical
and Computing Sciences
Tokyo Institute of Technology
Tokyo, Japan

Biography

Satoshi Matsuoka received his Bachelor’s, Master’s, and Ph.D. in Information Science from the University of Tokyo. After being a Research Associate and Lecturer for the university’s Information Science and Information Engineering, departments, he became an Associate Professor in the Dept. of Mathematical and CS. Five years later, he became a professor at the Global Scientific Information and Computing Center (GSIC) of Tokyo Institute of Technology (Tokyo Tech)—ranked 2nd in Japan and 22nd in the world in engineering and IT, according to the Times rankings. Matsuoka leads GSIC’s Research Infrastructure Division overseeing Tokyo Tech’s responsibilities as a national supercomputing center.

He leads Tokyo Tech’s TSUBAME supercomputer effort, named the fastest supercomputer in Asia-Pacific in June 2006 for three consecutive Top500s. He also heads a lab of 25 researchers and graduate students for the university’s Dept. of Mathematical and CS. From 2003–2008 he served as a sub-leader of the Japanese National Research Grid Initiative project—a \$100 million national project to create middleware and applications for next-generation Japanese Cyber-Science (Grid) infrastructure. He has published over 200 refereed publications, many in top-tier conferences and journals, and has won several awards including the prestigious JSPS Prize from the Japan Society for Promotion of Science in 2006 from Prince Akishinomiya. He has played key roles in a number of international conferences, including program chair of ACM OOPSLA 2002, and technical paper chair of IEEE/ACM Supercomputing 09, the latter drawing over 10,000 attendees worldwide.

He has also held various key positions in academic societies, including secretariat of ACM Japan Chapter. His research interests focus on software issues in peta/exaflop supercomputers and clouds.

Statement

Computing is now playing unprecedentedly dominant roles in advancement of our society, and in solving some of the key challenges that mankind faces today, such as global warming and pandemic outbreaks. However, another recent poll showed that many non-IT researchers considered IT to be a diminishing research area, to be rather stagnant in the next 20 years. I had been involved in various large ACM conferences, academic societies, as well as many national projects especially in supercomputing / e-Science where multiple IT technologies needed to be harnessed, as well as being extremely interdisciplinary in its relationship to other disciplines—based on such experience, I propose to focus my initial role as an at-large council member to be the following: (1) to elevate the status of ACM conferences to be acknowledged amongst all academic disciplines as a top-rate publication, and (2) to establish an improved framework in tightening relationship of ACM with sister information societies and related academic archival activities outside of the United States, particularly in Asia Pacific. The former will be a key to enhance the prestige of IT research in academia, and will be achieved by setting and publicizing more streamlined standards in the conference review process. The latter will be demonstrable in collaboration with Information Processing Society of Japan (IPSJ) as well as other Asian societies, along with involving Japan’s National Institute of Informatics that owns one of the world’s largest pan-discipline academic publication databases. Hopefully such activities will foster better recognition of our field as an indispensable and ongoing research discipline during the 21st century.

candidates for

MEMBERS AT LARGE

(7/1/10–6/30/14)

**SALIL VADHAN**

Professor of Computer Science
and Applied Mathematics
Director, Center for Research
on Computation and Society
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA, USA

Biography

Salil Vadhan is the Vicky Joseph Professor of Computer Science and Applied Mathematics and Director of the Center for Research on Computation and Society at the Harvard University School of Engineering and Applied Sciences.

Vadhan received an A.B. *summa cum laude* in Mathematics and Computer Science from Harvard College in 1995, a Certificate of Advanced Study in Mathematics with distinction from Cambridge University in 1996, and a Ph.D. in Applied Mathematics from MIT in 1999. He was an NSF Postdoctoral Fellow at MIT and the Institute for Advanced Study before joining the Harvard faculty in 2001. He has also been a Fellow at the Radcliffe Institute for Advanced Study (2003–2004) and a Miller Visiting Professor at UC Berkeley (2007–2008).

Vadhan's research interests in theoretical computer science include computational complexity, cryptography, and randomness in computation. His Ph.D. dissertation on zero-knowledge proofs received the ACM Doctoral Dissertation Award in 2000, and his work on expander graphs received the ACM-SIGACT/EATCS Gödel Prize in 2009. He has also received a Guggenheim Fellowship (2007), an ONR Young Investigator Award (2004), a Sloan Research Fellowship (2002), an NSF Early Career Development Award (2002), and a Phi Beta Kappa Award for Excellence in Teaching (2004).

Vadhan's service to ACM includes the ACM-SIGACT Committee for the Advancement of Theoretical Computer Science (since 2007), the program committee of the 39th ACM Symposium on Theory of Computing (STOC 2007) and chairing the program committee of STOC 2011. Beyond ACM, he has served on numerous other conference program committees, editorial boards, and grant review panels.

Statement

As Member-at-Large, I will work to ensure that ACM's policies and organization continue to support the mission of the academic and research communities to advance and disseminate our scientific understanding of computing. Two issues in which I am particularly interested are:

- Conveying the importance of long-term, fundamental research in computing to the public and the government. From my service on the ACM-SIGACT Committee for the Advancement of Theoretical Computer Science, I have experience in such efforts and have seen the positive impact that they can have.
- Enabling the widest possible dissemination of scholarly research. I would like to see ACM seriously consider how it can support and join the movement toward open-access publishing models, which are being embraced by universities (such as Harvard) and the research community.

From my work with Harvard's Center for Research on Computation and Society, I will also bring an appreciation for the social and legal dimensions of computing research, particularly with respect to privacy and security.

**FEI-YUE WANG**

Research Scientist and Director of the Key Lab of Complex Systems and Intelligence Science at Chinese Academy of Sciences
Professor and Dean of the School of Software Engineering
Xi'an Jiaotong University
Beijing, China

Biography

Fei-Yue Wang received his Ph.D. in Computer and Systems Engineering from Rensselaer Polytechnic Institute, Troy, New York in 1990. He joined the University of Arizona in 1990 and became a Professor and Director of Robotics Lab and Program in Advanced Research for Complex Systems. In 1999, he founded the Intelligent Control and Systems Engineering Center at Chinese Academy of Sciences (CAS) under the support of Outstanding Oversea Chinese Talents Program. Since 2002, he is the Director of Key Lab of Complex Systems and Intelligence Science at CAS. Currently, he is the Dean of School of Software Engineering, Xi'an Jiaotong University, and Vice President of Institute of Automation, CAS.

His major research interests include social computing, Web science, and intelligent control.

From 1995–2000, Wang was the Editor-in-Chief of *Int'l J. of Intelligent Control and Systems* and *Series in Intelligent Control and Intelligent Automation*. Currently, he is the EiC of *IEEE Intelligent Systems* and *IEEE Trans on ITS*. He has served as Chair of more than 20 IEEE, ACM, INFORMS conferences. He was the President of IEEE ITS Society from 2005–2007, Chinese Association for Science and Technology in 2005, American Zhu Kezhen Education Foundation from 2007–2008.

From 2006–2008, Wang was the Founding President of the ACM Beijing Chapter. Currently he is an ACM Council Member-at-Large and Vice President and Secretary-General of the Chinese Association of Automation.

Wang is member of Sigma Xi and an elected Fellow of IEEE, INCOSSE, IFAC, ASME, and AAAS. In 2007, he received the National Prize in Natural Sciences of China and was recognized as an ACM Distinguished Scientist.

Statement

This year marks a special equilibrium in my life. By the end of 2010, I will have spent exactly the same number of years living and working in China and the US. This valuable experience offers me a unique perspective to understand and appreciate the similarity and differences between the East and the West, especially in issues related to academic exchanges, the promotion and advancement of science and technology. If elected, my focus would be to broaden the membership base, the scope, the reach, and the impact of ACM in Eastern Asia. In particular, I will do my best to help ACM in its process of creating the ACM China Council to recognize and support Chinese ACM members and ACM activities in China.

After the establishment of ACM Europe and India Councils in 2009 and 2010 respectively, an ACM China Council should be an imperative and important step. China, with its sheer numbers of students, engineers, researchers, and educators in computing professionals, will provide a huge opportunity for ACM's future development. However, as I had advocated at the ACM Council Meeting last October, we should try our best to get computing professionals inside China to be involved in the process of creating the ACM China Council and avoid the impression of imposing an ACM organization from outside. I will work hard to help ACM recruit new members in China and assist them to organize more local chapters, student chapters, and SIG chapters before launching the ACM China Council. I believe this approach is critical for a solid foundation and healthy growth of ACM in China and extremely helpful to the long term benefit of ACM.

Finally, social computing is the current focus of my research interest and I will make a significant effort to promote this emerging and important interdisciplinary field within ACM.

Economic and Business Dimensions Cloud Computing and Electricity: Beyond the Utility Model

Assessing the strengths, weaknesses, and general applicability of the computing-as-utility business model.

BUSINESSES RELY NO less on electricity than on IT. Yet corporations don't need a "Chief Electricity Officer" and a staff of highly trained professionals to manage and integrate electricity into their businesses. Does the historical adoption of electricity offer a useful analogy for today's innovations in cloud computing?

While the utility model offers some insights, we must go beyond this simple analogy to understand cloud computing's real challenges and opportunities. Technical issues of innovation, scale, and geography will confront managers who attempt to take advantage of offsite resources. In addition, business model challenges related to complementarity, interoperability, and security will make it difficult for a stable cloud market to emerge. An overly simplistic reliance on the utility model risks blinding us to the real opportunities and challenges of cloud computing.

Cloud Computing and the Electricity Model

Definitions for cloud computing vary. From a practitioner standpoint: "Cloud computing is on-demand access to virtualized IT resources that are housed outside of your own data center, shared by others, simple to

An overly simplistic reliance on the utility model risks blinding us to the real opportunities and challenges of cloud computing.

use, paid for via subscription, and accessed over the Web." From an academic perspective: "Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services. ... The data center hardware and software is what we will call a cloud. When a cloud is made available in a pay-as-you-go manner to the public, we call it a public cloud; the service being sold is utility computing."¹

Both definitions imply or explicitly use the "utility" model that embeds the logic of water supply, electrical grids, or sewage systems. This model is ubiquitous. While it has important strengths, it also has major weaknesses.

Hardware providers introduced the language of "utility" computing into the market. But perhaps the most rigorous and vigorous assertion of the electricity model comes from Nicholas Carr, an independent blogger in his recent book, *The Big Switch*: "At a purely eco-

conomic level, the similarities between electricity and information technology are even more striking. Both are what economists call general-purpose technologies. ... General-purpose technologies, or GPTs, are best thought of not as discrete tools but as platforms on which many different tools, or applications, can be constructed. ... Once it becomes possible to provide the technology centrally, large-scale utility suppliers arise to displace the private providers. It may take decades for companies to abandon their proprietary supply operations and all the investment they represent. But in the end the savings offered by utilities become too compelling to resist, even for the largest enterprises. The grid wins."⁴

Strengths of the Utility Model

Carr correctly highlights the concept of a general-purpose technology. This class of technology has historically been the greatest driver of productivity growth in modern economies. They not only contribute directly, but also by catalyzing myriad complementary innovations.³ For electricity, this includes the electric lighting, motors, and machinery. For IT, this includes transaction processing, ERP, online commerce and myriad other applications and even business model innovations.

Some of the economies of scale and cost savings of cloud computing are also akin to those in electricity generation. Through statistical multiplexing, centralized infrastructure can run at higher utilization than many forms of distributed server deployment. One system administrator, for example, can tend over 1,000 servers in a very large data center, while his or her equivalent in a medium-sized data center typically manages approximately 140.⁷

By moving data centers closer to energy production, cloud computing creates additional cost savings. It is far cheaper to move photons over the fiber-optic backbone of the Internet than it is to transmit electrons over our power grid. These savings are captured when data centers are located near low-cost power sources like the hydroelectric dams of the northwest U.S.

Along with its strengths, however, the electric utility analogy also has three technical weaknesses and three business model weaknesses.

Technical Weaknesses of the Utility Model

The Pace of Innovation. The pace of innovation in electricity generation and distribution happens on the scale of decades or centuries.⁸ In contrast, Moore's Law is measured in months. In 1976, the basic computational power of a \$200 iPod would have cost one billion dollars, while the full set of capabilities would have been impossible to replicate at any price, much less in a shirt pocket. Managing innovative and rapidly changing systems requires the attention of skilled, creative people, even when the innovations are creat-

consistency and scalability at the same time. The problem of scalable data storage in the cloud with an API as rich as SQL makes it difficult for high-volume, mission-critical transaction systems to run in cloud environments.

Meanwhile, companies of a certain size can get the best of both worlds by deploying private clouds. Intel, for example, is consolidating its data centers from more than 100 eventually down to about 10. In 2008 the total fell to 75, with cost savings of \$95 million. According to Intel's co-CIO Diane Bryant, 85% of Intel's servers support engineering computation, and those servers run at 90%



ed by others, unlike managing stable technologies.

The Limits of Scale. The rapid availability of additional server instances is a central benefit of cloud computing, but it has its limits. In the first place, parallel problems are only a subset of difficult computing tasks: some problems and processes must be attacked with other architectures of processing, memory, and storage, so simply renting more nodes will not help. Secondly, many business applications rely on consistent transactions supported by RDBMS. The CAP Theorem says one cannot have

utilization—a combination of strategic importance and operational performance that would negate any arguments for shifting that load to a cloud vendor. Ironically, even as the utility model is being touted for computing, the highly centralized approach is becoming less effective for electricity itself: an emerging distributed power generation system features smaller nodes running micro-hydro, wind, micro-turbines and fuel cells. What's more, many enterprises do in fact generate their own electricity or steam, for the same reasons they will continue to keep certain classes of

IT in house: reliability, strategic advantage, or cost visibility.

Latency: Distance is Not Dead. One of the few immutable laws of physics is the speed of light. As a result, latency remains a formidable challenge. In the network realm, the demands for nearly instantaneous execution of machine-to-machine stock trades has led financial services firms to locate their data centers as physically close to stock exchanges as possible. The read/write limits of magnetic disks can only drop so far, but increased speed comes at the cost of capacity: big disks are slow, and fast disks are small. For many classes of applications, performance, convenience, and security considerations will dictate that computing be local. Moving data centers away from their customers may save on electricity costs, but those savings are often outweighed by the costs of latency.

Beyond Electricity: The Business Model of the Cloud

Important as the technical differences are between electricity and cloud computing, the business model differences are even more profound.

Complementarities and Co-invention. Like electricity, IT is a general-purpose technology. This means that critical benefits come from the co-inventions that the basic technology makes possible. It took 30 to 40 years for the full benefits of electricity to redound to America's factories.⁵ Initially, assembly lines and production processes were not redesigned to take advantages of electricity: large central steam engines were simply replaced with large electric motors, and then hooked up to the same old crankshafts and cogs. Only with the reinvention of the production process was the potential of electrification realized. Today, electricity has matured to become a relative commodity. In contrast, computing is still in the midst of an explosion of innovation and co-invention.² Firms that simply replace corporate resources with cloud computing, while changing nothing else, are doomed to miss the full benefits of the new technology.

The opportunities, and risks, from IT-enabled business model innovation and organizational redesigns are reshaping entire industries.³ For instance, Apple's transition from a perpetual license model to the pay-per-use

If the utility model were adequate, the challenges to cloud computing could be solved with electricity-like solutions—but they cannot.

iTunes store helped it quadruple revenues in four years. The tight integration between Apple's ERP system and the billing engine handling some 10 million sales per day would have been difficult, if not impossible, in the cloud.

Lock-in and Interoperability. Lock-in issues with electricity were addressed long ago by regulation of monopolies, then later by legal separation of generation from transmission and the creation of market structures. Markets work because electrons are fungible. The rotary converter that enabled interconnection of different generating technologies in the 1890s has no analog for the customer of multiple cloud vendors, and won't anytime soon. For enterprise computing to behave like line voltage will require radically different management of data than what is on anyone's technology roadmap.

Perhaps most critically, bits of information are not electrons. Depending on the application, its engineering, and its intended use, cloud offerings will not be interchangeable across cloud providers. Put more simply, the business processes supported by enterprise computing are not motors or light bulbs.

Security. The security concerns with cloud computing have no electricity analog. No regulatory or law enforcement body will audit a company's electrons, but processes related to customer data, trade secrets, and classified government information are all subject to stringent requirements and standards of auditability. The typically shared and dynamic resources of cloud computing (including CPU, networking, and so forth) reduce control for

the user and pose severe new security issues not encountered by on-premise computing behind firewalls.

Conclusion

If the utility model were adequate, the challenges to cloud computing could be solved with electricity-like solutions—but they cannot. The reality is that cloud computing cannot achieve the plug-and-play simplicity of electricity, at least, not as long as the pace of innovation, both within cloud computing itself, and in the myriad applications and business models it enables, continues at such a rapid pace. While electric utilities are held up as models of simplicity and stability, even this industry is not immune from the transformative power of IT.^{8,9} Innovations like the "smart grid" are triggering fundamental changes at a pace not seen since the early days of electrification.

The real strength of cloud computing is that it is a catalyst for more innovation. In fact, as cloud computing continues to become cheaper and more ubiquitous, the opportunities for combinatorial innovation will only grow. It is true that this inevitably requires more creativity and skill from IT and business executives. In the end, this not something to be avoided. It should be welcomed and embraced. ■

References

1. Armbrust, M. et al. A view of cloud computing. *Commun. ACM* 53, 4 (Apr. 2010), 50–58.
2. Bresnahan, T., Greenstein, S., Brownstone, D. and Flamm, K. Technical progress and co-invention in computing and in the uses of computers. *Brookings Papers on Economic Activity—Microeconomics* (1996), 1–83.
3. Brynjolfsson, E. and Saunders, A. *Wired for Innovation: How IT is Reshaping the Economy*. MIT Press, Cambridge, MA, 2010.
4. Carr, N. *The Big Switch: Rewiring the World, from Edison to Google*. Norton, New York, 2008.
5. David, P. The dynamo and the computer: An historical perspective on the modern productivity paradox. *American Economic Review* 80, 2 (1990), 355–361.
6. Foley, J. Plug into the cloud. *InformationWeek* (Sept. 28, 2008).
7. Hamilton, J. Internet-scale service efficiency. In *Proceedings of the Large-Scale Distributed Systems and Middleware (LADIS) Workshop*, (Sept. 2008).
8. Hughes, T. *Networks of Power: Electrification in Western Society, 1880–1930*. Johns Hopkins University Press, Baltimore, MD, 1983.
9. Waltz, D. and King, J. *Information Technology and America's Energy Future*. Computing Research Association White Paper, Washington, D.C., 2009.

Erik Brynjolfsson (erikb@mit.edu) is a professor at the MIT Sloan School and the director of the MIT Center for Digital Business in Cambridge, MA.

Paul Hofmann (paul.hofmann@sap.com) is a vice president at SAP Labs in Palo Alto, CA.

John Jordan (jmj13@smeal.psu.edu) is a senior lecturer in the Smeal College of Business at Penn State University.

Copyright held by author.



Education

How to Make Progress in Computing Education

Improving the research base for computing education requires securing competitive funding commitments.

E DUCATION IS THE economic issue of the 21st century. Driven by global trade and a technologically connected, always-on global work force, countries understand they must innovate to succeed in the new business environment. A winning innovation policy is tricky to define, but it is clear it starts with education—and it starts early.

In the U.S., policymakers seem to have heard the message. There is a national urgency to improve K-12 education, and, in particular, ensure students have a strong grasp of science, technology, engineering, and mathematics (STEM) education. The Department of Education is pouring an unprecedented hundreds of billions of dollars into states to improve schools, help teachers, and support students. They want to know this money is helping. If you listen closely, you hear leaders from the Secretary of the Education to members of Congress talking about the need for “evidence-based” reforms. Where does this evidence come from? Largely, it comes from measurement tools developed by education researchers.

At the same time, the computing community sees a national urgency to reform K-12 computer science education. As computing transforms society for the digital age, students need to be able to think computationally about the world to succeed in life. How do students really learn rigorous computing concepts? We need research to tell us.



Computing is a relatively new discipline with a small education research base and limited assessments. Those responsible for making policy decisions in K-12 are interested in adopting curriculum in schools where you can assess how it is improving student learning. They are also interested in focusing resources on the “core” that students must know. Rigorous computing courses, if they exist, aren’t typically in the “core.” This leads to a chicken-and-egg problem for K-12 computer science, where you can’t really measure

how students learn without putting it in schools, but schools aren’t interested in it until you can measure it.

We need to break this cycle and one aspect is improving the research base for computing education.

It isn’t enough to rely on general education research. We need research specific to computing—a form of *domain-specific education research*. General education research helps us understand (for example) how students learn and how schools best facilitate learning. Domain-specific education

research answers questions that are unique to the domain. Mathematics education researchers help us determine what preschoolers ought to know so they succeed later at multi-digit arithmetic (and how to remediate missing skills early, before they impede students' progress). Physics education researchers know why students have trouble understanding velocity and acceleration, and they have identified the visualizations and activities that can enhance learning.

Computing education research is necessary for us to improve our teaching of computer science. Researchers in computing education can tell us how students understand parallel algorithms, what kind of visualizations help with understanding data structures (and how to use them), and how to measure understanding about computing that goes beyond any single language. Computing education researchers help us understand why students do not pursue computing as a career, and how to recruit, engage, and motivate more (and more diverse) students.

But we are the new kids on the school block. The National Council of Teachers of Mathematics was founded in 1920. The National Association for Research in Science Teaching started in 1928. In comparison, ACM's Special Interest Group in CS Education (SIGCSE) is only 40 years old, and ACM started the Computer Science Teach-

ers Association (CSTA) six years ago. SIGCSE's research conference, International Computing Education Research (ICER) Workshop, is only in its fifth year.

Being relatively new puts us at a disadvantage when seeking competitive funding. Imagine that you are seeking funding in a general education research program, in competition with proposals in mathematics education and science education.

► Which proposals have great evaluation plans that will demonstrate the value for the investment? Mathematics and science education can point to scores of reliable, valid measures of learning that they can use. In computing education, there is no reliable, valid measure of introductory learning that isn't tied to a specific programming language. Overall, there are few standard measures of computing learning.

► Which proposals will lead to more students achieving learning objectives, identified by standards at the state or national level? Such standards and objectives exist for mathematics and science, but rarely for computer science in the U.S.

Some countries *do* fund research in computing education. There are strong research programs in computing education in Israel (Technion and Open University), England (at the University of Kent at Canterbury, for example), Germany (Free University Berlin),

Sweden (Uppsala University), and Finland (at University of Joensuu). These research programs are investments in IT competitiveness in those countries.

The State of Computing Education Research Funding in the U.S.

How about in the U.S.? Things are much more dismal, particularly for the K-12 level. The National Science Foundation (NSF) is primarily responsible for funding education *research*,^a which comes two directorates: Computer and Information Sciences and Engineering (CISE) and Education and Human Resources (EHR). We examine CISE first.

CISE has had two programs—CISE Pathways to Revitalized Undergraduate Computing Education (CPATH) and Broadening Participation in Computing (BPC)—with a focus on education. However, as of this writing CISE announced that it is combining these programs into a broader program. This new vision would look at the entire pipeline but with special focus in two areas:

► moving earlier into the pipeline with specific engagements in middle/high school to bring computational thinking/computer science concepts into this space; and

► widening the program to be inclusive for all populations, built around a theme that “computing is for everyone.”

It would also add a specific education research component that would seek to build education research capacity at the university level and to provide a greater understanding of how children come to understand computing concepts. No one knows exactly what this new program will look like until the solicitation comes out, which CISE is saying will happen in the summer of 2010. It is expected the new program will be funded at about \$20 million, which is similar to the combined amount for CPATH and BPC.

a We did not do a detailed review of grants from the Department of Education's research arm—The Institute of Education Science—as this institute appears to be focused on general education research. A cursory review did not find any grants focused specifically on computing research. Further, other programs run by the Department of Education are primarily focused on funding state and local education agencies to put resources directly into the schools.

Results of NSF "Fastlane" abstracts summary analysis.

Program	CS participation rate	Number of CS hits	Number of Proposals
ITEST	9%	18	202
Grad Teaching Fellows K-12	6%	20	316
Gender in Sci/Engineering	4%	8	187
Research and Evaluation on Education in Science and Engineering (REESE)	3%	11	413
DR K-12	2%	6	289
Robert Noyce* Teacher Scholarship Program	1%	4	282
Math and Science Partnerships (MSP)	0%	0	150
Total	4%	67	1839

* Noyce is not a research program; rather it is a program that prepares K-12 teachers in specific STEM disciplines. Computing may do poorly in this program because of serious teacher certification issues for computer science teachers, which have been explored in a report by the Computer Science Teachers Association: <http://csta.acm.org/ComputerScienceTeacherCertification/sub/TeacherCertificationRequi.html>

These are likely positive steps toward addressing clear gaps in the field. But it will likely be a series of small steps until the community can start leveraging other parts of NSF. Compared to the relatively small CISE budget for education, EHR has over \$850 million for education research, which is where we need to turn our attention. Not all of this funding goes into education research, but in looking where Congress is investing federal education research money, it is clear they are looking to EHR for those answers. EHR funds both higher education and K-12 research programs through various programs.

The program that probably does the most for higher-education computer science is the Course, Curriculum, and Laboratory Improvement (CCLI) program. It seeks to improve undergraduate education across STEM through proposals on how interventions work and how they get disseminated, and funds the development of new assessment techniques and instruments. It funds applied research that informs interventions, and doesn't fund computing education research that helps us develop new theory about how people come to understand computing.

The state of computing education research and teacher support at the K-12 level is more complicated. There are several relevant EHR funding programs. ACM staff analyzed abstract summaries from NSF's "Fast-lane" within EHR to better understand where computer science education research is funded or where computer science teacher support existed. The scope of EHR programs was limited to: *funded* proposals that had a significant K-12 focus, or those that prepared or continually trained K-12 teachers. Abstracts are only a brief representation of the plan, so the analysis tended to be more inclusive—"close" was good enough. However, the analysis specifically excluded grants that were primarily focused on getting computing into the classroom or getting teachers prepared to use computing in the classroom.

The results of the analysis appear in the table here. Of the 1,839 proposals funded across seven programs, only 67 (4%) had an explicit computer science

It isn't enough to rely on general education research. We need research specific to computing—a form of domain-specific education research.

component. Our analysis of abstracts could not tell us which of these projects had any kind of research component, nor where the research informed our understanding of learning computing specifically.

Regardless of the limitation of the analysis, it is clear—there is far too little computing research or teacher support being done by the key agency charged by the federal government for understanding how to improve STEM education.

Making Progress in Computing Education

Funding is important. Funding allows researchers to make progress on problems that are a priority. Funding is recognition that a particular discipline or strategy is worthwhile. Funding can create a virtuous circle, where funded work attracts more funded work. Lack of funding creates a vicious circle, when the lack of theory and of assessment prevents other projects from being funded.

The computing education research is concerned with how to sustain interest and progress in the research community. Few of the U.S.-based presenters at the International Computing Education Research Workshop have NSF funding to work in computing education research. Those that have received NSF funding do the computing education research component on the side. Few Ph.D. students focus on computing education, and those that do focus on this area rarely obtain faculty slots in research institutions. Working in an area without explicit funding programs is dangerous for an untenured assistant professor at a research institution.

Funding is particularly important to bootstrap a field. Computing education research seems to be in a vicious cycle. As a community we need to take some basic steps to break the cycle:

- Learn what NSF programs are available and aggressively go after funding. NSF CCLI Program Officers regularly offer SIGCSE workshops walking through the various NSF programs that are available for education research to catalyze proposals.

- Sit on NSF review panels when asked, *particularly* in EHR. There should be a computing voice at these review panels. The best way to learn what gets funded (and how to write a fundable proposal) is to sit on these panels.

- Encourage fundamental research in computing education. As teachers of computing, we want to know what language, IDE, book, and curriculum works best in our classroom. We also need the theory that helps us make these choices, and the assessment that gives us data on what works best. We want students to start successfully and to develop expertise and skill.

- Look for opportunities to work with other domain-specific education groups. Mathematics education, for example, has funding sources, and computing education research could grow in combined projects.

- We must stand together. Reform proposals supported by many institutions carry weight. Shared goals and strategies lead to proposals that reviewers can support.

Computing education research is an important investment in innovation and competitiveness. If the U.S. wants to remain the world leader in computing, it ought to be making that investment and the community needs to aggressively go after funding. Other countries are making that investment. Growing computing education research in the U.S. would improve teaching and learning in computing nationally and inform the research community internationally. □

Cameron Wilson (wilson_c@hq.acm.org) is director of the ACM U.S. Public Policy Office in Washington, D.C.

Mark Guzdial (guzdial@cc.gatech.edu) is a professor in the College of Computing at Georgia Institute of Technology in Atlanta, GA.

Copyright held by author.

Viewpoint

Can IT Lean Against the Wind?

Lessons from the global financial crisis.

THE SEPTEMBER 2009 *Communications* Editor's Letter "The Financial Meltdown and Computing" by Moshe Vardi suggested a link between the financial crisis of 2008 and computing. He is correct to suggest this connection. Information technology (IT) has enabled ever-increased speed and global reach for financial products. Financial institutions in the U.S. and elsewhere have created and deployed complex, structured financial instruments. At the peak of the bull market, the seemingly endless promise of fancy financial products drove the markets to new heights. What went wrong, and what role did IT play? This column cannot provide all the answers, but it offers some recent history and a lesson worth remembering.

The Role of IT in Financial Markets

Before the financial meltdown of late 2008 two important events provided a glimpse into the role of IT in the financial markets. The first was the September 11, 2001 terrorist attack on the World Trade Center. The attack destroyed a prominent symbol of Wall Street. It also destabilized the financial clearing and settlement systems of major banks located nearby. Foreign exchange settlements in U.S. currency collapsed, which could have created a global financial calamity. However, in part because of remedies applied during the Y2K crisis, emergency back-up systems in the IT infrastructure prevented the worst. Within three hours, disas-



ter recovery systems located abroad took over from New York to handle all U.S. currency transactions, and clearing and settlement for U.S. currency was up and running.

The second event was the London terrorist attack of July 7, 2005 that resulted in a partial shutdown of the London Stock Exchange (LSE). The LSE systems were unprepared to handle the flood of automatically generated trades intended to contain losses to distributed financial institutions. The LSE therefore asked member institutions to shut down their algorithmic "black box" trading systems, and created instead a fast market for non-binding, indicative pricing to make up the difference. Member institutions complied, shutting down their black box systems long enough for

the LSE systems to begin handling non-algorithmic trades properly.

These examples prove that highly complex, IT-based financial systems can be remarkably reliable. Problems in key centers such as New York or London were handled without global crisis. Backup systems on other continents were brought online to prevent financial disaster. In both cases, threats originated from outside the system, and the system responded well. The financial meltdown of 2008 was due to threats inside the system. What we now call toxic assets were in effect sleepers and Trojan horses, embedded in the system by the system's participants. Intrusion detection systems could not alert the risk managers because there were no intrusions. IT people had never imagined

that financial industry professionals in investment banks, brokerages, pension funds, and other organizations lacked the tools to treat simultaneous crises from operational, credit, and market risks through an integrated risk assessment. This was not the kind of crisis IT specialists planned for.

Liquidity Crisis

The collapse of Northern Rock, the U.K.'s fifth-largest bank, was the visible warning of the debacle to come. When Northern Rock was taken over by the British government in February 2008, no one considered the problems as related to IT. The crisis of Northern Rock was not big enough for that purpose. However, when Lehman Brothers failed on September 15, 2008 the role of IT in the debacle became clear. When Lehman Brothers faltered, financial institutions around the world were forced to reevaluate their risk exposure almost instantly. All of their IT systems were built on presumptions of an orderly flow of prudent business transactions; no one had imagined that the transactions themselves might be the problem. Lehman Brothers was an investment bank, an essential intermediary in global credit markets. Many banks had hundreds of millions of dollars queued up in payments to Lehman Brothers when the news broke. There were no IT routines in place to stop such transactions once they were initiated.

When it became clear that money was about to go into a black hole, the IT specialists in the banks did the only thing they could do: they pulled the plug on the IT infrastructure, thereby halting all operations. Banks around the world became risky partners simply because no one knew who was risky and who was not. All transactions were stopped and cash flow came to a halt. This was the dreaded "liquidity crisis" that is still being discussed widely. The only way banks could avoid sending good money after bad was to disconnect the IT systems from the global financial networks. Within hours, the lightning-fast global financial system had slowed to the speed of the pre-computer era. The effects were pervasive, hitting even the smallest financial institutions in the most remote corners of the Earth.

This crisis was not caused by IT, but an imbalance in IT infrastructure

It was not possible to assess risks of transactions as they occurred, so financial industry experts simply assumed the transactions were OK.

played a major role in its origin. The problem was a discrepancy between two essential capabilities: the ability to execute transactions and the ability to comprehend the implications of the transactions being executed. IT departments within financial institutions were able to deliver "millisecond information flows" for real-time processing of transactions. However, they could not support counterparty credit risk calculations at speeds to match the transactions. It was not possible to assess risks of transactions as they occurred, so financial industry experts simply assumed the transactions were OK. A few experts making risky assumptions might be protected if the vast majority of experts are executing due diligence and evaluating risks carefully. The few benefit from the equivalent of "herd immunity" in vaccinations against disease. When all of the experts assume the transactions are OK, serious trouble can follow.

Credit risk calculations require a lot of work. The data for them must be gathered from many—sometimes several hundred—data warehouses. Data in such systems is often inconsistent and subject to quality control problems. During crises expert analysts often face absurdly simple but debilitating problems, such as trying to determine what the headers in their data sets mean, or trying to deduce which financial partners have provided given data. It seems difficult to believe that such data problems were allowed to continue even as IT sped up transactions to light speed. But as it often happens with IT, different parts of the IT ecology develop at different speeds.

Calendar of Events

May 17–19

Computing Frontiers Conference,
Bertinoro, Italy,
Sponsored: SIGMICRO,
Contact: Nancy M Amato,
Email: amato@cs.tamu.edu

May 26–28

The International Conference on Advanced Visual Interfaces,
Rome, Italy,
Contact: Giuseppe Santucci,
Email: santucci@dis.uniroma1.it

June 1–4

23rd International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems,
Cordoba, Spain,
Contact: Moonis Ali,
Email: ma04@txstate.edu

June 1–4

International Conference on Supercomputing,
Tsukuba, Japan,
Contact: Taisuke Boku,
Email: taisuke@cs.tsukuba.ac.jp

June 3–4

The 20th International Workshop on Network and Operating Systems Support for Digital Audio and Video,
Amsterdam, Netherlands,
Contact: Dick Bulterman,
Email: dick.bulterman@cw.nl

June 4–5

Design Science Research in Information Systems and Technologies,
St. Gallen, Switzerland,
Contact: Purao Sandeep,
Email: sandeep-purao@psu.edu

June 5–7

International Symposium on Memory Management,
Toronto, ON Canada,
Contact: Vitek Jan,
Email: jv@cs.purdue.edu

June 5–6

ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering,
Toronto, ON Canada,
Contact: Atanas Rountev,
Email: rountev@cse.ohio-state.edu

ACM Transactions on Reconfigurable Technology and Systems



This quarterly publication is a peer-reviewed and archival journal that covers reconfigurable technology, systems, and applications on reconfigurable computers. Topics include all levels of reconfigurable system abstractions and all aspects of reconfigurable technology including platforms, programming environments and application successes.

www.acm.org/trets
www.acm.org/subscribe



Association for
Computing Machinery

Data Exchange Standardization

IT specialists might be surprised to learn that there are no standardized data exchange formats for traded asset classes. Some financial experts say it is difficult or even impossible to develop data exchange standards that cover all elements needed for sound risk assessments. The financial market is highly product driven, with extremely short development cycles. Standardization of data exchange formats might never catch up with what is being traded. But, as every seasoned IT professional realizes, such standardization must be *part of the product*. Otherwise, functions that require standardization, such as real-time counterparty credit risk calculation, might never catch up with the risks being taken. Legislators and regulators seeking to tame the financial markets must look at these matters systematically. Mandatory standardization of data exchange formats based on emerging schemes (for example, Financial product Markup Language, FpML) might have to be developed for different asset classes so that a common understanding of offerings and risks is possible with sufficient speed to accompany financial product offerings.

At present, financial firms cannot assess risk exposure in real time. They collect the necessary data and do the math during nighttime batch processing operations that can last hours. It would be a huge improvement if there were nothing more than systems to support initial heuristic risk assessment, but this might not be enough to avoid problems such as those of 2008. It might be necessary to slow transactions down until risk assessment can occur at the same speed the transactions occur.

Former chairman of the U.S. Federal Reserve Alan Greenspan has said that bank risk managers are more knowledgeable than government bank regulators, and that regulators cannot “lean against the wind” to dampen economic swings. This might be correct, but bank risk managers need the right systems to do their jobs. The IT systems used to support assessment of counterparty credit risk are not as mature as transaction systems, especially for integrated assessment of operational, credit, and market risks. Individual desks at an institu-

It might be necessary to slow transactions down until risk assessment can occur at the same speed the transactions occur.

tion might do a good job at evaluating risks for their department, but they lack the global enterprise perspective that is vital to a global financial industry. This must change. Financial institutions are looking for improvements to data management, risk evaluation algorithms, and simulation systems, not because they are forced to do so by regulation, but because these are essential to their survival. The crisis has shaken confidence in the ability of IT systems to support the risk assessment at the heart of financial system operation, regulation, and market transparency. However, only by improving IT systems to support such assessment can the global financial industry move forward.

Perhaps financial regulators should not lean against the wind, but improved IT systems might play a vital role by helping bank risk managers do their jobs more effectively. IT professionals, working closely with colleagues from other business departments, can create industrywide, canonical data exchange standards to help with management of risk by improving data quality across organizations and borders. In this way, IT might lean against the wind of threats to global financial markets by enabling mature and embedded analytics that must influence decisions in financial institutions. Pulling the plug was a poor response to the crisis of 2008; the next time it might not work at all. 

Roman Beck (rbeck@wiwi.uni-frankfurt.de) is an assistant professor with the E-Finance Laboratory in the Department for Economics and Business Administration at Goethe University Frankfurt, Germany.

Copyright held by author.

Viewpoint

Is Mobile Email Addiction Overlooked?

Studying the prevalence of mobile email addiction and the associated possible implications for organizations.

HE REACTS PROMPTLY to every sound from his BlackBerry. He checks his BlackBerry continuously, can't even think of missing a single message, and responds aggressively if you distract him. Once he tried to get rid of his BlackBerry but could not because he became depressed. His family, friends, and the entire world cease to exist when an email message arrives. It looks like he lives only in order to check and respond to his email. Sounds familiar? We know a person like this. Do you?

Over the past several years, tens of millions of users have acquired BlackBerry, iPhone, or other devices supporting email applications. In many cases, users received these devices from their organizations. A major driver of the spread of mobile email is its ubiquity and convenience—people may check their email and respond from anywhere anytime. On the one hand, mobile email helps employees connect with their organizations and increase productivity. Employees can become more aware of and responsive to organizational, customer, and peer needs. Indeed, organizational benefits resulting from mobile email usage are unarguable. On the other hand, some individuals may become addicted to mobile email. Certainly, anecdotal evidence supports the existence of mobile email addiction; for example, the term “crackberry” was coined for describing the addictive nature of such technologies.



Mobile email addiction is a form of non-substance addiction that involves excessive interaction with both a mobile technology (mobile device) and the content (electronic communication) under conditions of psychological dependency. It can be viewed as a special type of a broader Internet addiction, as the latter concept involves excessive email messaging (but also other behaviors such as excessive gaming and sexual preoccupation).² The ubiquitous nature of mobile email technologies can facilitate and augment excessive email

preoccupation, which is no longer restricted to one's office but rather could be done anytime and from anywhere.

Symptoms

Mobile email addiction may be manifested through many symptoms. When using mobile email, an addicted person may notice the activity dominates his or her thoughts and behaviors, offers a thrill or relief, and it is difficult to control or quit this behavior. It conflicts with other people or tasks, and causes negative emotions when interrupted.

The symptoms of this addiction may dramatically affect an addict's well-being.⁵ First, social quality of life may be compromised as people may complain about one's preoccupation with mobile email. Some users may react negatively when others interrupt their email tasks but later feel ashamed about their overuse of mobile email. Some prefer working with their mobile email rather than interacting with family and friends, even in intimate situations. Second, the addicts' family relationships may be affected when they neglect family and home duties. Third, mobile email can become a "mental safe haven" for escaping from daily realities. Individuals may keep themselves busy with mobile email to avoid doing other more mundane tasks.

Perspectives

There are two conflicting points of view on the addictive nature of contemporary technologies. The proponents of this so-called addiction suggest that some users could demonstrate problematic usage behaviors that may be considered pathological and require treatment—hence technology addiction is a psychiatric disorder that merits research, legislation, and formalization. Over the past decade, a number of terms, such as Internet addiction disorder, computer addiction, technology addiction, virtual society addiction, pathological use, and problematic use were coined.¹¹ In support of this argument, it has been shown that technology addiction goes beyond the notion of mere overuse⁴ or high engagement.³ It has also been demonstrated that these problematic usage behaviors may lead to a range of negative consequences including depression, mood alteration, loneliness, isolation, and reduced impulse control; many experience work, family, social, interpersonal, health, and financial problems.

The opponents of the technology addiction concept argue that the aforementioned claims are unwarranted, that problematic use of technology exists only in very narrow contexts, such as gambling and emailing, and that technology overuse is a result of other preexisting mental disorders (such as reduced impulse control).¹⁰ As it stands, this is the prevalent medical view in

North America. It is argued that the border between technology addiction and other mental issues is blurred because 86% of identified Internet addiction cases have some other mental disorders present.² As a result, the current version of *Diagnostic and Statistical Manual of Mental Disorders* (DSM-IV-TR), which includes the formal list of mental disorders that is used by American psychologists and psychiatrists, does not recognize any types of technology addictions. Despite being lobbied by doctors, academics, and research centers, the American Medical Association chose not to consider video game addiction and Internet addiction serious medical disorders.⁷ First, many believe the term addiction may be used with respect to chemical substances only. Second, an established definition, set of symptoms, and diagnosis criteria are missing. For example, the description of the Internet addiction disorder was based on pathological gambling documented in DSM, and critics say most of the technology overuse criteria may be found under the existing DSM categories, such as obsession, compulsion, impulse control, depression, or anxiety. Thus, it is unlikely that such addictions will appear in DSM-V that is tentatively scheduled for publication in 2012.

Effects

From a *Communications* reader perspective, however, the mobile email (so-called) addiction phenomenon deserves special attention, because it may have negative consequences for users, their families, and their organizations. Moreover, mobile email addiction is distinct from most other types of technology addictions. For example, Inter-

The ubiquitous nature of mobile email technologies can facilitate and augment excessive email preoccupation.

net or video game addicts have made a personal decision to repeatedly engage in a potentially dangerous pathological behavior. In contrast, it is an organization that in most cases provides mobile email technology, pays for it, and requires its usage even beyond regular business hours. Therefore, the behavior in question in organizational settings can be facilitated and encouraged by an authoritative third party rather than by the users. As a result, addicts may hold organizations legally responsible, and companies may face potential liability issues.⁹ For a court to recognize damages resulting from an excessive use of mobile email, five categories must be established:

- **Duty**—whether an organization owes a duty to a person who became addicted. On the one hand, the addiction may be self-inflicted when the individual voluntarily engaged in technology overuse. On the other hand, the addict may argue that the organization owed the addict a duty to prevent the addictive nature that was facilitated, required, and encouraged by the organization. The standard of duty the organization owes the employees must also be determined.

- **Breach of Duty**—whether an organization deviates from the standard of care that a responsible employer would follow. For example, if the usage of mobile email beyond regular working hours has become an irrevocable part of organizational culture, the employer had encouraged this practice, and did nothing to prevent potential addiction; a reasonable argument can be made in the courtroom.

- **Proximate Cause**—whether mobile email addiction and its symptoms resulted from system overuse for work-related purposes. In other words, a clear causal link between mobile email usage and negative consequences must be established.

- **Actual Cause**—the employee must establish that but for the organization requiring the use of mobile email, the employee would not be addicted.

- **Damages**—whether the mobile email addict suffered from substantial physical or psychological damages. For instance, an employee may claim that his or her addiction behavior caused serious marital problems such as divorce. In fact, it is the family members

of mobile email users who mostly complain about the issue.

There is no clear evidence to conclude whether organizations should be liable when their employees develop mobile email addiction and suffer from related symptoms. As the society and social norms change, so do the laws. Currently, a number of BlackBerry addicts have already filed lawsuits against their employers; in some cases, organizations decided to settle out of court to avoid negative publicity.⁶ Employers, therefore, should be prepared for various scenarios.

In addition to legal issues, mobile email addiction may potentially have other negative consequences for organizations. It is reasonable to assume that employees who are addicted to their mobile email suffer from mood alterations, feelings of work overload, and negative effects on their familial lives. Thus, they may be likely to feel less satisfied with their jobs, and ultimately voluntarily leave their organizations. But how prevalent is the mobile email addiction phenomenon? To what extent is this addiction associated with voluntary turnover intentions (intentions to look for a job at a different company)?

To explore these issues, we surveyed 241 current mobile email users from three North American organizations. The questionnaire asked users 19 questions about the frequency in which they incur six technology addiction symptoms (based on the Internet Addiction Disorder Scale^{5,11}), and four questions that measured turnover intentions. The included symptoms were: compromised social quality of life due to overuse of mobile email, compromised individual quality of life, compensatory usage (using mobile email instead of doing other things that need to be done), compromised career, compromised time control (using mobile email longer than intended), and excitatory usage of mobile email (such as blocking disturbing thoughts with thoughts about mobile email). Reported frequencies ranged from once a year or less to every day.

In order to assess the levels of addiction, two scenarios were developed. Under the conservative scenario, it was assumed that at least four out of the six symptoms should be reported with a high frequency of at least sev-

Mobile email addiction may be considered more prevalent than other technology addictions if we follow the liberal criteria standard.

eral times a month. In this case, only 6.2% of the sample may be classified as pathologically addicted. Under a more liberal scenario, in which at least three symptoms are needed with a moderate frequency of at least once a month, 17.4% of the sample may be considered addicted. These results demonstrate that some individuals, between 6% and 17%, may meet mobile email addiction criteria. Furthermore, a correlation of 0.15 (significant at $p < 0.05$) between the addiction scores and turnover intentions was observed.

Taken together, these results demonstrate that mobile email addiction may be a fairly common phenomenon, and that it can be associated with negative organizational consequences such as turnover. Should we be concerned? These percentages can translate into millions of users who present worrisome levels of mobile email addiction disorder, or for those who oppose the technology addiction concept, high levels of mere technology overuse.

It is interesting to see how mobile email addiction compares to other technology addictions. Particularly, does the ubiquity of mobile email make it more addictive? While there are no known comparable samples for which we have technology addiction scores, it has been reported that 13.7% of Chinese adolescent Internet users meet addiction criteria,² and that 7.5% of a sample of teenagers has been diagnosed with severe psychological dependency on the Internet.⁸ The percentage of mobile email addicts in our sample is in line with technology addiction levels reported in the studies mentioned here. In fact, mobile email addiction

may be considered more prevalent than other technology addictions if we follow the liberal criteria scenario. Nevertheless, because different measures were used with different populations and with different addiction-cutoff values, we cannot firmly conclude whether the ubiquity of mobile email increases its addictiveness compared to that of other technologies, such as the Internet. This important distinction warrants future studies.

Given the empirically demonstrated potential pervasiveness of mobile email addiction, and extrapolating from existing frameworks for preventing Internet abuse and overuse¹—it is suggested that organizations employing mobile email monitor the extent to which their employees utilize this technology for early detection of addiction, control the usage as necessary (limit usage hours), educate employees and managers about addiction risks when distributing mobile email devices, and develop appropriate policies for mitigating future legal risks. **G**

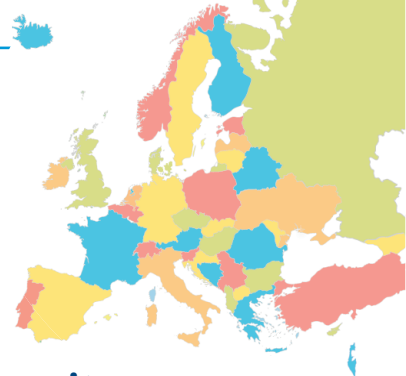
References

1. Anandarajan, M. and Simmers, C.A. *Managing Web Usage in the Workplace: A Social, Ethical and Legal Perspective*. IIRM Press Hershey, PA, 2003.
2. Block, J.J. Issues for DSM-V: Internet addiction. *American Journal of Psychiatry* 165, 3 (Mar. 2008), 306–307.
3. Charlton, J.P. and Danforth, I.D.W. Distinguishing addiction and high engagement in the context of online game playing. *Computers In Human Behavior* 23, 3 (Mar. 2007), 1531–1548.
4. Davis, R.A. A cognitive-behavioral model of pathological Internet use. *Computers in Human Behavior* 17, 2 (Feb. 2001), 187–195.
5. Ferraro, G., Caci, B., D'Amico, A., and DiBlasi, M. Internet addiction disorder: An Italian study. *Cyberpsychology & Behavior* 10, 2 (Feb. 2007), 170–175.
6. Goodchild, S. and Hodgson, M. BlackBerry addicts: Why the workers who can't switch off are suing their employers. *The Independent*; <http://www.independent.co.uk>
7. Grohol, J.M. Video games no addiction for now *PsychOnline*; <http://www.psychcentral.com>
8. Hur, M.H. Demographic, habitual, and socioeconomic determinants of Internet addiction disorder: An empirical study of Korean teenagers. *Cyberpsychology & Behavior* 9, 5 (May 2006), 514–525.
9. Kakabadse, N., Porter, G. and Vance, D. Addicted to technology. *Business Strategy Review* 18, 4 (Apr. 2007), 81–85.
10. Yellowlees, P.M. and Marks, S. Problematic Internet use or Internet addiction? *Computers in Human Behavior* 23, 3 (Mar. 2007), 1447–1453.
11. Young, K.S. *Caught in the Net: How to Recognize the Signs of Internet Addiction—And a Winning Strategy for Recovery*. Wiley, New York, 1998.

Ofir Turel (oturel@fullerton.edu) is an associate professor of Information Systems and Decision Sciences at California State University, Fullerton.

Alexander Serenko (aserenko@lakeheadu.ca) is an associate professor of MIS at Lakehead University in Ontario, Canada.

Copyright held by author.



ACM Europe Council

Serving the European Computing Science Community

ACM announces the ACM Europe Council: 16 leading European computer scientists from academia and industry to spearhead expansion of ACM's high-quality technical activities, conferences and services in Europe.

- ◆ Holding high-quality ACM conferences in Europe
- ◆ Expanding ACM chapters
- ◆ Strong co-operation with other European scientific societies in computing



"Our goal is to share ACM's vast array of valued resources and services on a global scale. We want to discover the work and welcome the talent from all corners of the computing arena so that we are better positioned to appreciate the key issues and challenges within Europe's academic, research, and professional computing communities, and respond accordingly," says **Professor Dame Wendy Hall, U. of Southampton (UK), ACM President.**

Supporting the European Computing Community

ACM Europe aims to strengthen the European computing community at large, through its members, chapters, sponsored conferences and symposia. Together with other scientific societies, it helps make the public and decision makers aware of technical, educational, and social issues related to computing.

A European Perspective Within ACM

The ACM Europe Council brings a unique European perspective inside ACM and helps increase visibility of ACM across Europe, through:

- ◆ Participation of Europeans throughout ACM
- ◆ Representation of European work in ACM Awards and Advanced Membership Grades

ACM – World's Largest Educational and Scientific Computing Society

Since 50 years, ACM strengthens the computing profession's collective voice through strong leadership, promotion of the highest standards, and recognition of technical excellence. ACM supports the professional growth of its members by providing opportunities for life-long learning, career development, and professional networking.



Fabrizio Gagliardi, ACM Europe Chair and Director of External Research Programs, Microsoft Research Europe says, *"By strengthening ACM's ties in the region and raising awareness of its many benefits and resources with the public and European decision-makers, we can play an active role in the critical technical, educational, and social issues that surround the computing community."*

ACM is present in Europe with 15 000 members and 41 chapters. 23 ACM Turing Awards and other major ACM awards have gone to individuals in Europe. 215 Europeans received an Advanced Membership Grade since 2004.

The ACM Europe Council represents European computer scientists. Contact us at: **acmeurope@acm.org** or **<http://europe.acm.org/>**

<http://europe.acm.org>



Association for
Computing Machinery

Advancing Computing as a Science & Profession

Computer Museum Series

Great Computing Museums of the World, Part Two

The second of a two-part series highlighting several of the world's museums dedicated to preserving, exhibiting, and elucidating computing history.

SOME OF THE science and technology museums around the world are devoted to science discovery—to teaching their visitors, especially children, about the principles of science and technology. Other science and technology museums are more focused on the history and cultural significance of particular scientific discoveries and technological inventions. Some museums include a blend of the two functions.

This is the second installment of a two-part *Communications* series featuring five of world's greatest computing museums. These museums have been chosen for their contributions to the history and culture mission, though most of them have some elements of the science discovery mission as well. There are perhaps hundreds of small and not-so-small museums around the world either devoted entirely to computing or at least having significant computing exhibits. The museums highlighted in this series have been selected because of the large size of their exhibits, the importance and quality of the artifacts shown, and the quality of their interpretations.

An exhibit is not simply a collection of artifacts; it includes signage and other accompanying information (films, lectures, guided tours) that help to interpret the artifacts and set them in context. Each of the exhibits described in this series is the result of years of hu-



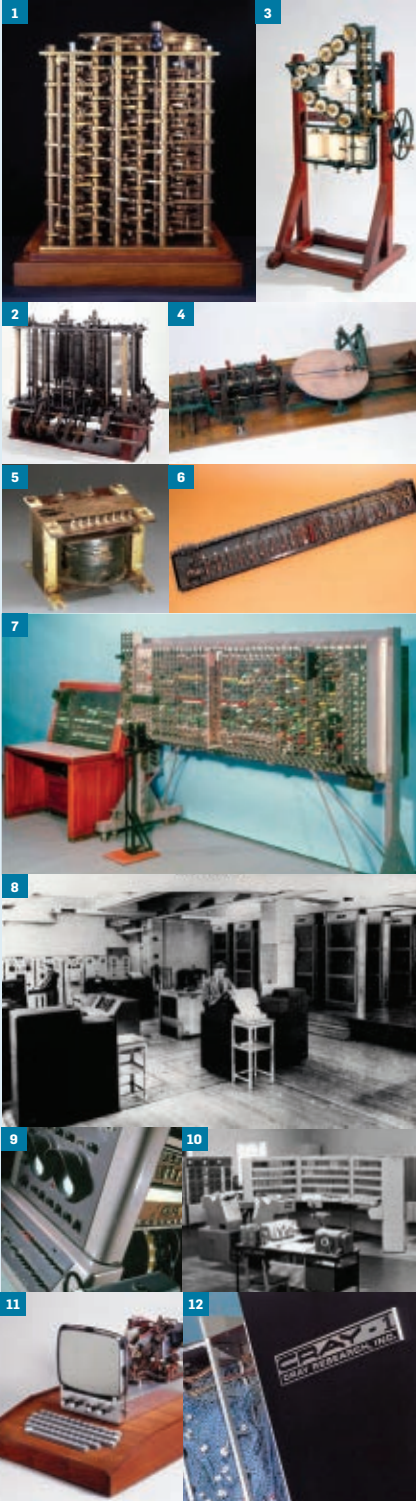
A selection of the U.S. National Museum of American History PC collection.

man labor in preparation: designing the exhibit, selecting and securing the artifacts, and giving them the right interpretation. This work has been carried out by some of the best historians of science and technology, who work in these museums collecting artifacts and the associated information and documentation about them, answering queries from all kinds of people about their collections and about the science and its history, undertaking scholarly research, preparing educational materials, and doing much more. The exhibits are only one facet of what these museums do.

The museums featured in this column are the Science Museum in London, the Deutsches Museum in Munich, and the U.S. National Museum of American History in Washington, D.C. (The first part of series appeared in the January 2010 issue.) We hope you enjoy the accounts of these museums and that these stories will whet your appetite to explore the museums' Web sites and to visit the museums in person.

William Aspray (bill@school.utexas.edu) is Bill and Lewis Suit Professor of Information Technologies at the University of Texas, Austin and a *Communications Viewpoints* section board member.

Highlights of the London Science Museum's Computing Collection



1. Difference Engine No. 1: This trial portion of the Difference Engine was used by Babbage as a demonstration machine.

2. Analytical Engine Mill, 1834–1871: Babbage's design possessed all the essential logical features of the modern general-purpose computer.

3. Lord Kelvin's tide predicting machine, 1876: By cranking the handle, the machine calculates the harbor's tide patterns for up to a year in only four hours.

4. Hartree and Porter differential analyzer, 1934: Built using Meccano at a cost of just £20, the model was based on the differential analyzer built by Vannevar Bush at MIT, 1930.

5. Parts of Colossus, 1943: From the code-breaking machine used during the Second World War at Bletchley Park.

6. Unit from ENIAC, 1946: Developed at the University of Pennsylvania, the machine laid the foundations for high-speed digital computing.

7. Pilot ACE, 1950: The first machine to embody the theoretical ideas of a general-purpose computer by mathematician Alan Turing.

8. Unit from LEO 1, 1951: The first business computer used by the Lyons teashops.

9. Ferranti Pegasus Computer, 1959: Currently the oldest working electronic computer in the world.

10. BESM-6 supercomputer, 1966: The only known example of a Russian supercomputer in the West.

11. Apple I, 1976: This first Apple computer was a kit machine for home assembly.

12. Cray I-A, 1976: The last operating Cray I-A in the world, the machine employed a cylindrical design to reduce the need for wiring.

The Science Museum in London

Tilly Blyth

The Science Museum's collections in London form an enduring record of scientific, technological, and medical change since the 18th century. The Science Museum has its origins in the Great Exhibition of 1851. Initially part of the South Kensington Museum, the impetus was to promote the new "industrial arts" (what we might now call art, craft, science, design, and engineering).

The museum's science collections were enriched in 1876 with the acquisition of a large number of scientific instruments, and the engineering collections grew through the absorption of the Patent Office Museum in 1883. The museum attracts over 2.6 million visitors a year, and is world renowned for its historic collections, awe-inspiring galleries, and inspirational exhibitions.

The Science Museum's collections celebrate computing as one of the most important technologies of our time, both in its own right, but also as an underpinning and enabling technology for so many other industries (see <http://www.sciencemuseum.org.uk/>).

The museum's computing collections are perhaps most well known for containing the seminal objects and material legacy of the mathematician and inventor Charles Babbage (1791–1871). In Victorian Britain printed mathematical tables were used by navigators, architects, engineers, mathematicians and bankers, but these tables were calculated by human clerks (literally called calculators) and they were quite often riddled with errors. Charles Babbage became interested in mechanizing the production of these tables and he developed a series of diagrams and prototypes to enable him to explore his ideas.

Babbage designed two types of engine, Difference Engines and Analytical Engines. The Difference Engines are calculators that work on a mathematical principle of "a method of finite differences," which allow polynomial equations to be calculated using pure addition. In contrast, The Analytical Engines mark a progression toward a general-purpose machine. They are one of the startling intellectual feats of the 19th century

and tangibly support Babbage's reputation as a computing pioneer.

Examples of both types of machine are on display, including Babbage's original trial piece for the Difference Engine No. 1, a portion of the mill of the Analytical Engine, and the first completed engine, the Difference Engine No. 2, built by the museum in 1991 for the 200 centenary of Babbage's birth. With over 4,000 parts, and weighing five tons, the machine can calculate numbers up to 31 digits long.

The Science Museum's displays also reflect Britain's role at the forefront of computing research in the 1940s and as central to the creation of a new global computing industry in the 1950s. During this period, Britain created the digital, electronic computer, the code-breaking Colossus machine (1943), the first stored-program computer, the Manchester Baby (1948), and the first business computer, the Lyons LEO (1951). Artifacts from all of these machines are on display in the Computing Gallery.

The Science Museum's galleries also showcase three important complete computers from this pioneering industry: the Pilot ACE computer (1950), embodying the original ideas of the mathematician Alan Turing and his conceptual discovery of the general-purpose machine; the Ferranti Pegasus (1959), which was fast and reliable and is now the oldest working electronic computer in the world; and ERNIE (1957), the first random-number generator for the national Premium Bonds that used a hybrid of valves and transistors and generated physical random events through a series of neon gas diodes.

Contemporary machines on display in our *Making the Modern World* gallery include the Cray 1-A supercomputer (circa 1976) and an Apple 1 (1976). Displayed side by side, the research machine (the Cray was installed at Aldermaston Atomic Weapons Establishment in England) and the Apple kit home computer built by enthusiasts provide a strong message about the shift to personal computing during the late 1970s.

Since its founding, the museum has always had a remit to display contemporary science and technology alongside its historical collections. In doing

The Science Museum's collections celebrate computing as one of the most important technologies of our time.

so it has presented new computing technologies using art and interactive elements, through a computer arts booth (1975) and exhibitions like *The Challenge of the Chip* (1980). Today, visitors flock to view *Listening Post* by Ben Rubin and Mark Hassan, an art installation that presents a dynamic and visually enticing portrait of Internet communication.

In addition to the public galleries, the Science Museum Library and Archives in London (see http://www.sciencemuseum.org.uk/about_us/about_the_museum/science_library.aspx) offers a world-class collection on the history, biography, and social context of science and technology.

The collections in Swindon offer original scientific and technical books and journals, alongside computing archives of important historical interest. These include the personal papers and technical drawings of Charles Babbage, Mike Woodger from the National Physical Laboratory, and Stanley Gill, who was significant in early U.K. computing policy.

The Archives Collection holds a range of computer literature, most notably through the ICL archive that contains significant information about the products and history of the merged companies that became ICL, including British Tabulating Machine, Powers-Samas, ICT, and English Electric.

The Science Museum Library collection also holds trade literature specific to a particular machine, installation, or company.

Tilly Blyth (tilly.blyth@sciencemuseum.org.uk) is the curator of computing and information at the Science Museum in London, U.K.

The Deutsches Museum

Hartmut Petzold

The Deutsches Museum is one of the world's biggest and oldest museums for science and technology, today with approximately 1.3 million visitors a year. Founded in 1903 as a "Museum of Masterpieces of Science and Technology," in 1906 Kaiser Wilhelm II placed the foundation stone for the new museum building on an island in the Isar river in the Bavarian capital Munich; the museum was completed and opened in 1925. The fundamental idea, which continues influencing the museum, involves presenting the newest "masterpieces" as a logical and historical extension of their precursors. From the very beginning this conception has been both criticized and also copied in many ways.

Planning the exhibition "Informatik" (English translation: computer science) began in 1983. At that time, there were only some experts knowing the Internet, and the personal computer was an expensive and special device. The "Informatik" exhibit opened in 1988; the initiator of the exhibition project was Professor Friedrich L. Bauer, who involved many university- and academic-based experts and collaborators. Bauer, as a mathematician and co-founder of computer science as a new academic discipline, had no difficulties presenting the historical instruments and machines as ancestors of the computer.

The museum's exhibition room borders on the exhibitions on microelectronics, telecommunication, and geodetics. It is subdivided into cabinet-like thematic units by numerous walls providing explanatory text. Therefore visitors do not typically initially notice that the middle of the hall is marked by Konrad Zuse's early program controlled calculators Z3 and Z4. The Z4's original parts are presented in the same condition as they were in 1955, when they were used at the ETH Zurich. The Z3, representing Zuse's priority as the inventor and builder of the first free programmable automatic calculator, working in 1941, and anticipating an essential part of John von Neumann's classical computer concept from 1945, is a reconstruction, built by Zuse himself from 1961–1962 (the original

machine had been destroyed by the bombs of World War II). Representing one of the highlights of the Deutsches Museum, the Z3 is still demonstrated to visitors. Located near the Dehomag/IBM plugboard programmed punched card tabulating machine D11, the pioneer machines of Konrad Zuse also mark the middle of the exhibition in a technohistorical sense. Zuses's machines separate the historical mathematical instruments, the mechanical calculators, and sequence controlled historical automatons in the first part of the exhibit from the spectrum of the big, middle, and small real computers in the second part of the exhibit.

One main issue is the separation of analog and digital calculating instruments and machines. The contrast is highlighted by red text tables for all explanations on analog technology, and by blue tables for all texts on digital technology. One of the outstanding early analog calculating instruments is the exceptionally big astrolabium, made by Erasmus Habermel in 1588. A showcase with different modern slide rulers, mainly used by engineers until the 1970s, is a logical follow-on. This part of the exhibition also includes a big and unique collection of mechanical planimeters, and some components of several big mechanical and electronic analog computers.

The blue-marked part with digital instruments and calculators begins with a small collection of different abaci. A historical sequence of selected mechanical digital desk calculators starts with Wilhelm Schickard's reconstructed machine, and is continued by replicas of the machines of Blaise Pascal and Gottfried Wilhelm Leibniz, all designed during the 17th century. Particularly valuable is an original calculator, made by Anton Braun and Philippe Vayringe and presumably completed around 1735. Beside these historical unique specimens one can view the broad spectrum of the mechanical calculators, industrially produced in series, beginning with C.X Thomas and ending with the sophisticated Friden-Model SRW 10. Many of the mechanical calculators were still used in the 1970s. A special cabinet on the cryptology is presented as a part of the digital calculating exhibit—here not with digits but with letters. Visitors find several coding machines, includ-

The fundamental idea involves presenting the newest “masterpieces” as a logical and historical extension of their precursors.

ing two Enigmas. Some historical sequence controlled automatons are also shown as a part of the prehistory of the computer, one of them the famous mechanical trumpeter, made by Friedrich Kaufmann in 1810.

In the second half of the exhibition with real computers, the gigantic central processing unit of the UNIVAC I Factronic with its mercury delay line memory is particularly representative. Visually it is overshadowed a bit by the so-called PERM computer (Programmgesteuerte Elektronische Rechanlage München) with its magnetic drum memory, which has been built at the Technical University Munich as a scientific project financed by the Deutsche Forschungsgemeinschaft (German Research Foundation) from 1950 until 1956. The UNIVAC computer, which was used at the Battelle-Institute at Frankfurt/Main between 1956 and 1963, represents the start of industrially produced computers in the U.S., whereas the PERM represents the beginning of research and development of electronic computers at German universities. Also interesting are some original parts of the U.S. pioneer computers ILLIAC and ORACLE. Several computers, developed and produced by the then West German industry are included in the gallery. With the exception of the vacuum tube machine Z22, produced by the Zuse KG in 1958, they all were designed on the base of discrete semiconductor elements.

Also presented is a Cray I from 1983, which until the end of Cold War in 1990 had its completeness checked for the COCOM-office (Coordinating Committee on Multilateral Export Con-

trols) by an employee of the company. Several office computers produced by the companies Nixdorf, Kienzle, and Siemag/Philips, as well as several very early personal computers and electronic pocket computers are displayed at the end of the exhibition.

More artifacts of the collection of historical calculating instruments, machines, and computers are stored in the depositions of the museum, mostly in other buildings. Fundamental for research in the history of computers and computer science particularly in Germany, but not only, are the documents collected in the archive of the Deutsches Museum. In the estates of some German computer pioneers like Konrad Zuse, Heinz Billing, and Joachim N. Lehmann researchers can find many written and printed documents. In addition, the archive includes a large collection of historical printed materials like catalogues and advertising leaflets from many computer-producing companies.

Hartmut Petzold (h.petzold@deutsches-museum.de) is the former curator for mathematical instruments, compilers, and time measurement at the Deutsches Museum in Munich, Germany.

U.S. National Museum of American History

David K. Allison

The U.S. National Museum of American History has collected and preserves more than three million artifacts that document the history of the American people. These range from the original “Star-Spangled Banner” and Abraham Lincoln’s top hat to Alexander Bell’s telephone prototypes. The museum’s collections form a mosaic of American life and comprise the nation’s greatest single collection of historical objects.

The Division of Information Technology and Communications is one of eight curatorial divisions in the museum. Collections in the Division include computing, mathematics, electricity, printing, graphic arts, photography, and numismatics. The Computing Collection has approximately 2,000 objects in the following categories: supercomputers and components; mainframe computers and components; minicomputers and components; microcomputers and components; electronic calculators; analog computers; computer

games; documentation; computer paraphernalia; and related devices. The related Electricity Collection includes electronic components; microchips; cellular telephones; and personal digital assistants. Printers are distributed among the Computer Collection, the Photography Collection, and the Printing Collection. Indeed, the process of digital convergence increasingly links all the collecting units in the Division.

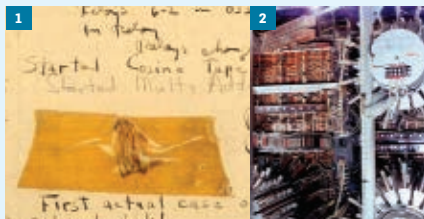
Further information about all of the components of the Division is available from the museum's Web site: <http://americanhistory.si.edu>. Among the materials there are transcripts of oral and video history interviews with leaders in American computing including J. Presper Eckert, Seymour Cray, Kenneth Olsen, Bill Gates, Steve Jobs, and Larry Ellison.

In addition to the holdings of the curatorial division, the museum's Archives Center has a number of computer-related collections (see <http://americanhistory.si.edu/archives>). Among its holdings are papers of Grace Murray Hopper, Ralph Baer, and Herb Grosch. Of particular interest are transcripts and other materials from a Computer Oral History project sponsored by the American Federation of Information Processing Societies in the 1960s and 1970s (see http://invention.smithsonian.org/resources/fa_comporalhist_index.aspx).

Although the computer collection is readily available to researchers, currently, little of it is on physical display at the museum. A major exhibition on the history of computing and communications, "Information Age: People, Information, and Technology" ran from 1990–2006, but has now closed. A new exhibition that will include coverage of digital computing is currently under development. Tentatively titled "American Enterprise," it is slated to be the Smithsonian's first comprehensive Smithsonian exhibition on the history of the American economy and to survey innovation in finance, manufacturing, agriculture, energy, information technology, and communications from the late 18th century to the present. 

David K. Allison (allisond@si.edu) is chairman of the information technology and communications department at the National Museum of American History in Washington, D.C.

Highlights of the U.S. National Museum of American History



1. The "First Computer Bug," 1947: The Smithsonian has the research notebook with what is reputedly the "first computer bug," a moth taken from a register in the Mark II computer at Harvard and taped into the notebook with the note "first actual case of bug being found."



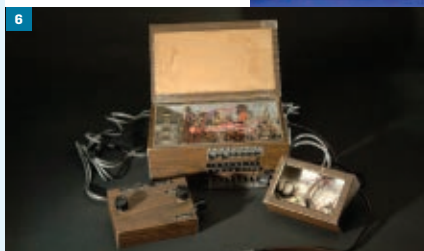
2. A UNIVAC I Console, 1951: In addition to the Console, the collection includes a mercury delay line memory unit and an arithmetic chassis unit.



3. The Institute for Advanced Study Computer, 1952: Developed at Princeton's Institute for Advanced Study in Princeton, NJ, the design for this computer was replicated in other early machines, such as MANIAC at Los Alamos and ILLIAC at the University of Illinois.

4. An IBM 650, 1954: This is an example of IBM's first mass-produced computer. The Smithsonian has a console unit and card reader/punch, plus documentation.

5. A Digital Equipment Corporation PDP-8 minicomputer, 1965: The collection includes both the processor and documentation.



6. The "Brown Box," 1967: A prototype for the first video game developed by inventor Ralph Baer.



7. A Xerox Alto, 1973: Developed at Xerox's Palo Alto Research Center, this device paved the way to graphical user interfaces and networked desktop computers.

8. Early personal computers, 1975–present: The Smithsonian collection includes a range of personal computers including several Altair 8800s, Apple IIs, Radio Shack TRS-80s, IBM PCs, an Apple Lisa, a Timex Sinclair, and one of Michael Dell's PC Limited computers, as well as more modern devices.



9. CIX Router, 1994: This Cisco Systems 7500 router was used between 1994 and 2001 as part of the first Commercial Internet Exchange. It was a private, membership organization that allowed networks to exchange Internet traffic directly, regardless of which network the customer obtained service from, at no additional charge for the traffic.

10. Deep Blue, 1997: The Smithsonian has one of the two towers of IBM's Deep Blue computer, which won the first regulation chess match against a world champion.

An essential technique used in emulator development is a useful addition to any programmer's toolbox.

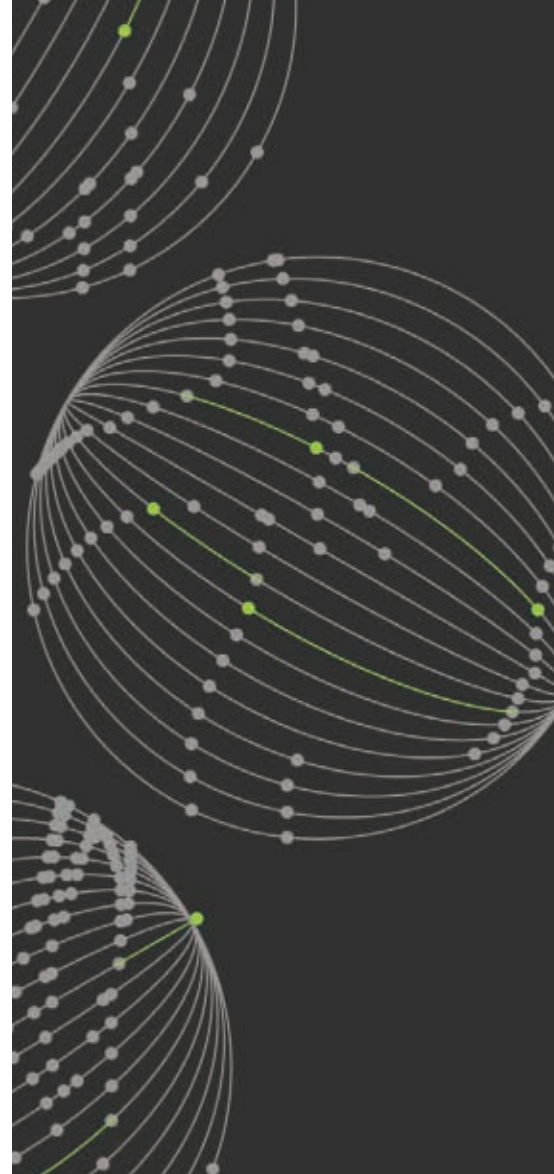
BY PETER PHILLIPS

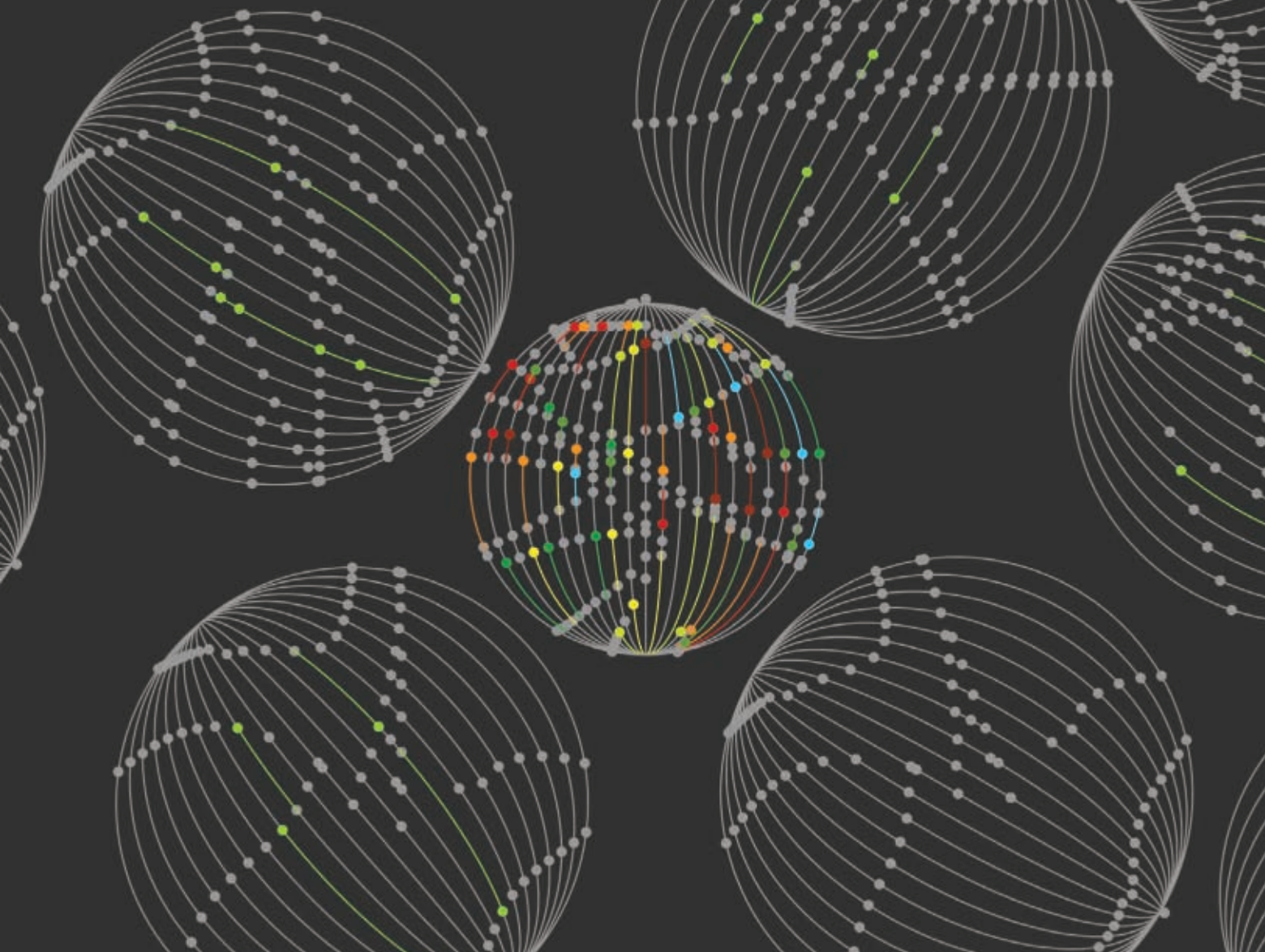
Enhanced Debugging with Traces

CREATING AN EMULATOR to run old programs is a difficult task. You need a thorough understanding of the target hardware and the correct functioning of the original programs that the emulator is to execute. In addition to being functionally correct, the emulator must hit a performance target of running the programs at their original real-time speed. Reaching these goals inevitably requires a considerable amount of debugging. The bugs—which for players of old arcade games are a vital part of recreating the original experience—are often subtle errors in the emulator itself but could also be a misunderstanding of the target hardware or an actual known bug in the original program. (It is also possible the binary data for the original program has become subtly corrupted or is not the version expected.) Solving these problems requires some unusual debugging techniques.

The debugger of the host system can be used for the target (emulated) system by setting a breakpoint in the CPU do-instruction loop and examining the variables that hold the contents of the emulated CPU registers and main memory. This works for small and easily localized problems such as failures in the self-test or initialization code but is not of much use for problems that happen at a later stage in the execution. At that point the organization of the program is not apparent. No debug symbol or source code is available to allow a high-level approach to the problem. More powerful techniques are required and are built into the emulator itself.

Obviously there is enormous variability in the kinds of bugs that may present themselves. Ignoring major execution path failures (for example, infinite loop, the program counter ending up in nonprogram memory), a typical failure would be noticed in





the video output of the target, shown in the accompanying figure. In this example a sprite's location is definitely incorrect. The top sprite is offset from being in the expected column and is at 13,11 instead of 24,11. We know the aliens always attack in columns, so something is wrong with the emulator. A pair of values in the emulator's memory state determines the sprite's position. This address would, in fact, be the location of a video hardware register that directly determines the displayed position. You can use the platform debugger to confirm the register values are, in fact, 13,11. Unfortunately, even if you know the values are wrong, finding the actual error is not easy.

The incorrect register values are the end of a chain of earlier calculations (see the accompanying sidebar "Screen Image Chain"). Finding the error involves working backward through the causal sequence to find where the mis-

calculation occurred. The debugger, however, shows us only the current state of the machine. If it is possible at all, reconstructing the event chain will involve a lot of detective work.

Traces to the Rescue

There may be an easy shortcut to solving the problem. If you are trying to create a faster emulator from an existing one that is known to work, then you can use the correct emulator to debug the incorrect one. The correct emulator can be modified to create a trace of input and CPU state, and the target emulator can read this trace file (see the accompanying sidebar "An Example 6502 Trace"). The input values are used instead of real input (joystick positions). After each `do-instruction` call, the current machine state is matched with the trace, and execution is halted on divergence. At this point you can use the host debugger to examine the problem in detail and

find the resolution.

What if, as is more likely the case, you don't have a reference trace? You still know the ultimate source of the error is when the incorrect *X* value 13 is placed into memory location 0xc000. You can search the trace file to find cases when this value is written (for example, search the `BUS` column for `C000w0d`). At this point you will see the incorrect value coming from a register, and looking just a few trace lines back, you will identify the address of the incorrect value in the `shadow[]` memory (the "Screen Image Chain" sidebar shows the path followed).

At this point you can apply the same steps with the (now) known address of the shadow memory to find the point in the trace where the *sprite* struct is modified. As you keep tracing backward, you will find either more memory locations to trace or sections of code that can be hand-verified to find the offending emulation error. It

Screen Image Chain

In bits of partial C code, here is how a sprite position is determined, starting from the actual hardware value and working backward. The C code is reverse engineered, as the emulator would execute machine instructions directly.

1. The image position of sprites is determined by hardware registers starting at 0xc000 (each register has a size of two bytes):

```
Sprite #0 0xc000 Xpos Ypos Shape Palette
Sprite #1 0xc008 Xpos Ypos Shape Palette
...
```

In this case, 0xc000 contains 0x000d 0x00b (corresponding to 13,11 on screen).

2. The registers are set at the vertical blank (60 times per second) frame by copying from an internal array.

```
struct sprite { unsigned short x, y, shape, palette; }
shadow[16];
...
void vblank_intr() { memcpy( (void*)0xc000, &shadow,
sizeof(shadow));
```

3. The shadow values are updated in an object update routine.

```
void Object::update_sprite() {
shadow[this->sprite_index].x = this->x + this->x_center;
shadow[this->sprite_index].y = this->y + this->y_center;
}
```

4. The object changes the position based on speed.

```
void Object::move() {
this->x += this->delta_x;
if (this->x > 128) { this->x -= this->delta_x; }
...
}
```

The fourth and final level is the first point of interest. Perhaps something is wrong with the emulator's implementation of the compare instruction.

certainly isn't as easy as using a reference, but there is a good chance of finding the error without constantly rerunning the emulator.

Using Traces

An emulator using traces can hunt down problems in a machine description, but traces can also be useful when the problem exists in the code itself and not the machine description. Using such a trace can make finding memory corruption problems much easier. Consider a particularly nasty kind of corruption: the corrupted data structure is owned by the runtime memory allocator implementation. For example, the program crashes inside of a call to `malloc()` as it traverses the free list. The actual crash is apparent in the debugger but the ultimate cause is not.

Suppose that the direct problem is that a particular pointer value at location 0x8004074 in the allocator's linked list has been modified. Instead of a plausible value, it contains 0x1—an invalid and unaligned memory reference. The standard debugger cannot be of much more help at this point, as it can reveal the invalid value but not when it became invalid. (You could use a break-on-write, but that won't work if the corrupted address changes between runs.)

Here is where the trace solves the problem. Search the trace backward for the last write to the memory location 0x8004074. You may find an entry such as:

```
0x80004074: write 0x1 PC=0x1072ab0
...
```

Translating the given program counter (0x1072ab0) to a location in the source code will immediately reveal a likely cause of the corruption. For example, it may now be clear that the array bounds were not correctly respected. An attempt to store a value of 1 in the array resulted in the memory corruption. Of course, as in the sprite example, the actual source of the corruption may need to be traced further back.

Tracing for All Programs?

Tracing is useful for debugging ordinary problems, but how to generate

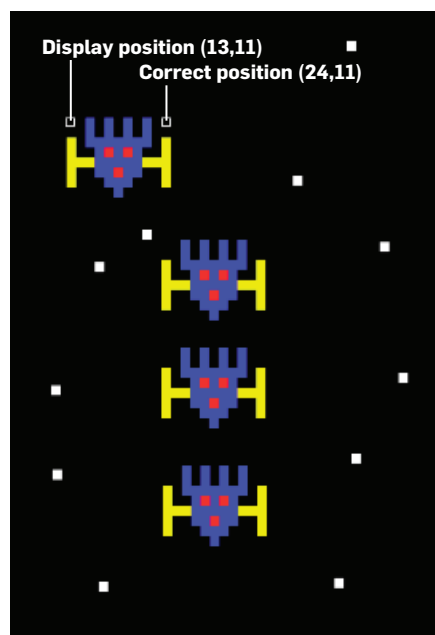
An Example 6502 Trace

The 8-bit 6502 microprocessor had only a few registers: A, X, Y, S, PC, and P. An actual system might also have a simple 8-bit input port for joystick positions. The trace file would be (hexadecimal) values of the registers and I/O port along with the current (symbolic) instruction:

PC	A	X	Y	S	P	IO	BUS	ASM
F010	00	01	02	F7	D1	01	F011r7A	lda #\$7A
F012	7A	01	02	F7	31	01	...	

In this example, the instruction at location F010 loads the value 7A into the A register, and the next line of the trace reflects the change. The high bit of the status register (named P on the 6502) is cleared as the value in A is not negative. A BUS column is included to show the reading or writing of data values onto the system bus.

There are a lot of possible ways to represent traces. For long traces, it may be more efficient simply to store entire machines states with a counter describing the number of instructions to execute before a new machine state is reached or an I/O value changes. The detailed trace can be reconstructed (as necessary) by running the emulator for N steps on a given machine state.



A sprite misplaced.

these traces isn't immediately apparent. Unlike the emulator, you do not control the (hardware) implementation of the microprocessor running your software.

One may object that such verbose logging is not particularly feasible for real programs. Although hard-drive storage continues to decrease in cost, I/O bandwidth does have limits. If storage is truly limited, there are a few other approaches. For example, the traces between snapshots could be stored only in RAM and not written to the log file. Traces could be recreated as necessary (by rerunning the target from a given state). Alternatively, only the most recent trace could be stored in main memory, providing quick access to scanning the trace if necessary.

The complete immutability of our CPU implementation is an elaborate illusion in many cases. All Java (and .NET) programs are actually running on virtual CPUs. Modifying these virtual machines to record trace information and state snapshots is conceivable—no hardware would have to be changed. It is merely a matter of convincing the owners of the virtual machine implementations to make the necessary changes.

Even native executables are a step removed from the real hardware. All modern operating systems enforce a separation between user and kernel modes of execution. A process is, in

fact, a virtual entity. The capability already exists to snapshot a process's execution state (Unix's venerable core dump). With some additional operating-system support, it is quite feasible to take these snapshot states and restore them as real processes that can continue their execution.

Indeed, the entire machine being used may be virtual. Products such as VMware and Parallels routinely operate with machine-state snapshots that include the entire state of user mode, kernel mode, device drivers, and even hard-disk drives.

Higher-Level Traces

Tracing memory accesses is helpful for programs such as assembler or C. Memory and CPU transfers correspond easily with actual source code, but it is more common to be working with languages a step removed from the machine where the C code is an interpreter of the actual language in use. In this case the low-level action of the code does not easily map to recognizable actions in the interpreted language.

The use of I/O is also more complicated in modern systems. Programs do not read and write directly to hardware I/O locations. Instead, device interaction is mediated through the operating system.

Tracing can be applied to these higher-level programs in an obvious fashion: change the trace to record higher-level events. The exact events to capture would depend on the kind of program. A GUI program may need to capture mouse, keyboard, and window events. A program that manipulates files would capture open/close and read/write operations. Code built on top of a database might log SQL statements and results.

A good trace differs from a simple log of events. The trace must provide enough information that the correctness of execution can be verified using only the trace. It should be possible to construct a *reference implementation* that can read the trace and automatically verify that the correct decisions are made. Experience with emulators suggests you might first code the reference implementation and use it to verify the production code. (You may avoid the premature optimization trap

and discover that the simplified reference implementation is a satisfactory solution.) Writing a reference might seem to involve as much effort as the production version, but there are always requirements that do not change the functional output. For example, the production code may require a lookup table to be persistent, whereas the reference can use a simpler in-memory hash table.

Conclusion

Adding snapshots, tracing, and playback to existing debugging environments would significantly reduce the time required to find and correct stubborn bugs. Low-level code is seeing some progress in this area; for some platforms, gdb has recently been given the ability to reverse execution. Since CPU operations are not reversible, this means there are now ways of capturing trace information for compiled programs. If the addition of saving and reloading snapshots were added, gdb could become a traceable debugger.

Detailed CPU state traces are extremely helpful in optimizing and debugging emulators, but the technique can be applied to ordinary programs as well. The method may be applied almost directly if a reference implementation is available for comparison. If this is not the case, traces are still useful for debugging nonlocal problems. The extra work of adding tracing facilities to your program will be rewarded in reduced debugging time. ■

Related articles on queue.acm.org

No Source Code? No Problem!

Peter Phillips and George Phillips
<http://queue.acm.org/detail.cfm?id=945155>

Debugging AJAX in Production

Eric Schrock
<http://queue.acm.org/detail.cfm?id=1515745>

Debugging in an Asynchronous World

Michael Donat
<http://queue.acm.org/detail.cfm?id=945134>

Peter Phillips received a bachelor's degree in computer science from the University of British Columbia. He has 15 years of experience in software development and has spent the past 10 years of that working on console games.

The key to synchronizing clocks over networks is taming delay variability.

BY JULIEN RIDOUX AND DARRYL VEITCH

Principles of Robust Timing over the Internet

EVERYONE, AND MOST everything, needs a clock, and computers are no exception. However, clocks tend to ultimately drift off, so it is necessary to bring them to heel periodically through synchronizing to some other reference clock of higher accuracy. An inexpensive and convenient way to do this is over a computer network.

Since the early days of the Internet, a system collectively known as NTP (Network Time Protocol) has been used to allow client computers, such as PCs, to connect to other computers (NTP servers) that have high-quality clocks installed in them. Through an exchange of packet timestamps transported in NTP-formatted packets over the network, the PC can use

the server clock to correct its own clock. As the NTP clock software, in particular the `ntpd` daemon, comes packaged with all major computer operating systems, including Mac OS, Windows, and Linux, it is a remarkably successful technology with a user base on the order of the global computer population.

Although the NTP system has operated well for general-purpose use for many years, both its accuracy and robustness are below what is achievable given the underlying hardware, and are inadequate for future challenges. One area where this is true is the telecommunications industry, which is busy replacing mobile base-station synchronous backhaul systems (which used to provide sub-microsecond hardware-based synchronization as a by-product) with inexpensive asynchronous Ethernet lines. Another is high-speed trading in the finance industry, where a direct relationship exists between reducing latencies between exchanges and trading centers, and the ability to exploit these for profit. Here accurate transaction timestamps are crucial. More generally, timing is of fundamental importance because, since the speed of light is finite, the latencies between network nodes are subject to hard constraints that will not be defeated by tomorrow's faster processors or bandwidth increases. What cannot be circumvented must be tightly managed, and this is impossible without precise synchronization.

The Clockwork

When the discussion turns to clocks, confusion is often not far behind. To avoid becoming lost in the clockwork, let's define some terms. By t we mean true time measured in seconds in a Newtonian universe, with the origin at some arbitrary time point. We say that a clock C reads $C(t)$ at true time t . Figure 1 shows what some example clocks read as (true) time goes on. The black clock $C_p(t)$ is perfect: $C_p(t) = t$, whereas the blue clock $C_s(t) = C_0 + (1 + x)t$ is out by C_0 when $t = 0$. In fact it keeps getting worse as it runs at a constant but overly



fast rate, the rate error being the *skew* x . The red clock $C_d(t) = t + E(t)$ is more realistic, its error $E(t)$ varies with time, and the clock is said to *drift*. Far from aimless, drift is in fact very closely related to temperature changes. Figure 6 (discussed later in greater detail) gives a close-up view of the drift $E(t)$ (black curve) for an unsynchronized clock in a Pentium PC over a two-day period.

At the heart of every clock is hardware. In PCs this is an oscillator found on the motherboard. The oscillator “ticks” at a nearly constant rate—in fact typically varying on average by only 0.1 ppm (parts per million). Hardware counters such as the HPET (High Performance Event Timer) count these ticks, giving the operating system access to a slowly drifting source of timing. From such a counter, a clock can be defined that, roughly speaking, converts counter tick units into seconds and adds a constant to set the time origin:

$$C(t) = C_0(t) + p(t) \cdot \text{HPET}(t)$$

where $p(t)$ is the (slowly time varying) period in seconds of the HPET counter. The role of the clock synchronization algorithm is to set and regularly update the values of C_0 and $p(t)$ to reduce the drift, or error $E(t) = C(t) - t$, as much as possible.

Enter the Internet. The sync algorithm has no way of correcting the drift of $C(t)$, inherited from the counter, without calling on some independent expert: a reference timing source, or master. Attaching extra hardware is possible but expensive, and is pointless unless it is reliable. A good-quality GPS with consistently good satellite visibility can cost many thousands of dollars by the time you have persuaded the building supervisor to mount it on the roof and run a cable to your office. Oh, and it will take you six months to make it happen—minimum. An atomic clock is even more expensive and still

requires GPS to avoid drift on weekly timescales and beyond. In contrast, synchronizing over the network can be done with no lead time, no dollars, and not much effort.

The Key Challenge: Delay Variability

Clock synchronization is trivial in principle. You ask the expert what the time is, he looks at his clock, tells you, and you set your watch. Simple. The problem is that each of these steps takes time, and on the Internet, these delays can be large, unpredictable, and highly variable. The key challenge—almost the only one—for a synchronization algorithm, is to be able to cope with these variations, to robustly and precisely negate the errors caused by delay variability.

The delays suffered by timing packets are generated in both the host computer and the network (and the server, but we will be generous and assume a

perfect reference clock). In the host, delays arise from the NIC (network interface card) behavior, operating-system process scheduling, and the architecture of the packet timestamping code, and are on the order of several tens of microseconds. In the network, they are caused by queuing in network elements such as IP routers and Layer 2 switches, and vary from a large fraction of a millisecond over a Gigabit Ethernet LAN to possibly hundreds of milliseconds

Figure 1. The time according to some simple clocks as a function of true time.

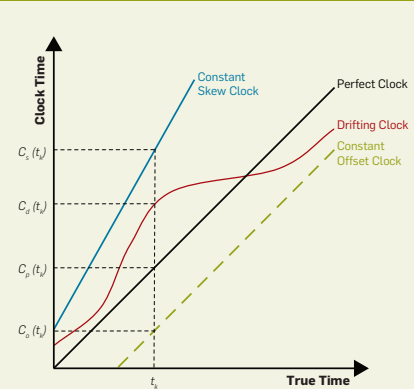


Figure 2. Bidirectional exchange of timing packets between host and server. A symmetric route would have $A = d^{\uparrow} - d^{\downarrow} = 0$, but that is not the case here. The RTT is $r = d^{\uparrow} + d^{\downarrow}$.

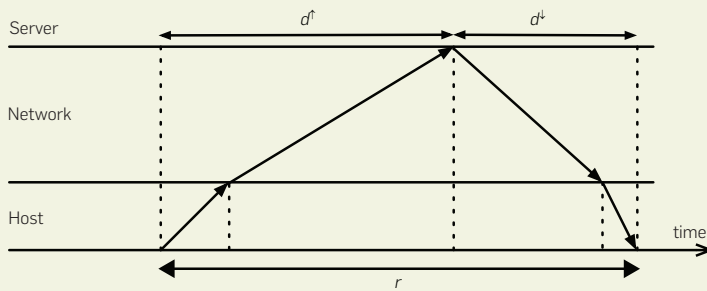
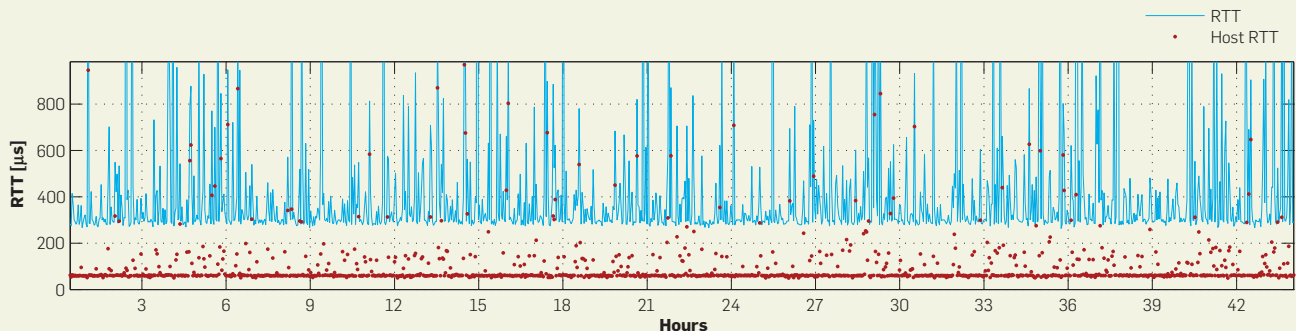


Figure 3. The RTT of timing packets over a congested LAN, and the host component separated out. Each component can be modeled as a minimum value plus a positive noise.



when network congestion is high and the server more than a few hops away.

In a bidirectional synchronization paradigm, which we focus on here (the alternative is to broadcast from the server only), timing packets are exchanged in both directions (see Figure 2). The OWD (one-way delay) in both the forward (host to server: $d^{\uparrow}$) and backward (server to host: $d^{\downarrow}$) directions have their own separate host and network components. Adding the OWDs in each direction yields the host→server→host RTT (round-trip time).

Figure 3 provides an example of the RTTs for timing packets when both the host and server are on a LAN, as well as the host component by itself. Each varies well above its minimum value, which translates to the addition of a large “noise” that the sync algorithm must somehow see through.

The Right Clock for the Job

The three main uses of a clock are to:

- Establish a strict order between events,
- Measure time durations between events, and

► Define a universally comparable and triggerable event label, the “time of day.”

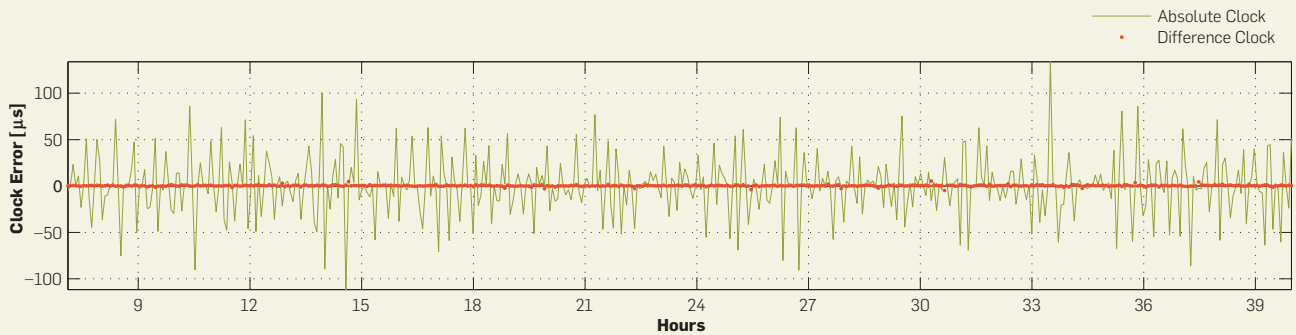
A perfect clock is ideally suited for each of these purposes, but in the real world, hardware and software limitations mean that the best performance is achieved by specialist clocks. The need to choose the right clock for the job is crucial and is not widely understood.

When one thinks of a clock, it is the third use that typically comes to mind: a clock that tells the *absolute* time. Synchronizing such a clock, however, is inherently difficult in the face of delay variability. As a result, the accuracy of an absolute clock $C_a(t)$ can be low, highly variable over time, and a slave to network conditions. Absolute clocks must also support the ability to be jump-reset, including backward. Though this may be rare, it makes them less than ideal for use 1.

A better choice for establishing temporal order is a well-behaved raw counter such as HPET(t), which increases monotonically. Equivalently, one can scale such a counter to form a simple causal clock calibrated in seconds: $C_c(t) = p_0 \cdot \text{HPET}(t)$ where p_0 is a constant that is never updated. Such a clock tells the wrong time and drifts, but does an excellent job of use 1 and is completely independent of network conditions, in fact not requiring server timestamps at all!

The errors of an absolute clock $C_a(t)$ also make it unsuitable for use 2. This is best seen through an example: if a counter has a rate error of 0.1 ppm (this is quite typical in a PC under reasonable temperature environments), then over a one-second interval the drift

Figure 4. In-kernel measurement of the one-second gap between pulses from a GPS receiver using a RADclock difference and RADclock absolute clock. Here the polling period to the server is 256 seconds. A single point per period is shown corresponding to the worst absolute error over the period.



amounts to an error of only $10^{-7} \cdot 1 = 0.1$ μs . On the other hand, errors in $C_a(t)$, and hence in the duration measurement, could be anything from several μs to several ms to larger than a second in extreme cases, depending on factors such as network congestion and algorithm robustness.

A better choice for the purpose of measuring time differences is to use a *difference* clock—that is, one that is not corrected for drift. A simple example is:

$$C_d(t) = C_0 + p_{av} \cdot \text{HPET}(t)$$

where C_0 is a constant (never updated), and p_{av} is an estimate of long-term average rate, which we can think of as a constant (we are simplifying here slightly for clarity; see Veitch et al.⁴ for details). Such a clock tells only approximately the right time and drifts, but does an excellent job of use 2 provided that p_{av} can be robustly and accurately estimated (which it can). It is reliant on server timestamps, but in a much less sensitive and more robust way than $C_a(t)$, as it takes direct advantage of the high stability (rate constant up to 0.1 ppm) of the local hardware.

Figure 4 compares the error in the time between pulses entering the serial port of a PC from a GPS receiver (nominally precisely one second apart) as measured by both an accurate absolute clock and an accurate difference clock. They are orders of magnitude different, even though the server was close by (minimum RTT of around 1ms). In fact, the operating system/serial port adds a noise of around 2 μs to these measurements and so actually swamps the error in $C_d(t)$: the differ-

ence clock is so good at the one-second scale that it is a serious challenge to measure its error!

A final crucial point: a difference clock works by strategically ignoring drift, but over longer durations the resulting error grows, and so the difference clock loses accuracy. A good rule of thumb is that the crossover occurs at $\tau = 1,000$ seconds. Above this scale durations should instead be calculated using an absolute clock. This crossover value can be calculated by finding the minimum in an *Allan Deviation* plot, which measures oscillator stability (variability as a function of timescale).

Time to Paradigm: Feedback or Feed-Forward?

Consider the following approach to synchronizing an absolute clock $C_a(t)$. Timing packets are timestamped in the host using C_a . These timestamps are compared with those taken by the server, and an assessment is made of the clock error. The rate of C_a is then slightly adjusted to bring that error to zero over time. This is an example of a *feedback* approach, because the previous round of clock corrections feeds directly back as inputs to the algorithm, because it is the clock C_a itself that is used to generate the host timestamps.

An alternative is to use the underlying counter to make *raw* packet timestamps (that is, counter readings) in the host. The clock error is then estimated based on these and the server timestamps, and “subtracted out” when the clock is read. This is a *feed-forward* approach, since errors are corrected based on post-processing outputs, and these are not themselves

fed back into the next round of inputs. In other words, the raw timestamps are independent of clock state. One could say that the hardware reality is kept in direct view rather than seeing it through algorithmic glasses.

The NTP synchronization algorithm is a feedback design. It uses a combination of PLL (phase-locked loop) and FLL (frequency locked loop) approaches to lock onto the rhythm of the server clock.² An example of a feed-forward design is that of the RADclock, a bidirectional, minimum RTT-filtered, feed-forward-based absolute and difference synchronization algorithm, that lies at the heart of our RADclock project at the University of Melbourne.⁴

The feedback approach has two significant disadvantages in the context of the Internet, where the delays are large and highly variable, and the feedback rate is low (the period between timing packets is typically between 64 and 1,024 seconds, since, for scalability, we cannot saturate the network with timing packets). The first disadvantage is that stability of classical control approaches such as PLL and FLL cannot be guaranteed. In other words, they can lose their lock if conditions aren’t nice enough, resulting in shifts to high-error modes that may last for long periods or even be permanent. Figure 5 gives an example of this, comparing the errors in `ntpd` and the absolute RADclock (sharing exactly the same timing packets to a server on an uncongested LAN) over two weeks. The feedback stability of `ntpd` is lost on a number of occasions, resulting in larger errors.

The second disadvantage is that difference clocks *cannot even be de-*

fixed in a feedback framework, so we lose their benefits, which include not only much higher accuracy, but also far higher robustness. It is worth noting that in networking, time differences are arguably more commonly used than absolute times. Delay jitter, RTTs, inter-arrival times, and code execution times are all examples of time differences that are best measured with a difference clock.

One disadvantage of a feed-forward approach is that it does not in itself guarantee that the clock never moves backward; however, a causality enforcing clock-read function can fix this without compromising the core design.

A Question of Support

The NTP system is the incumbent, supported by all major operating systems. Let us look at some kernel support issues, focusing on FreeBSD and Linux.

The system clock maintained by the kernel (which supplies the well-known `gettimeofday()` function call), and the kernel support for algorithms to discipline it, have historically been, and remain, closely tied to the needs of the `ntpd` daemon. In particular, the counter abstractions (`timecounter` in FreeBSD and `clocksource` in Linux), which give processes access to different hardware counters available in the system, fail to provide *direct* access to the raw counter values. Instead, these are accessible only via timestamps taken by the system clock that uses the counters under the hood. In other words, the kernel APIs support the feedback paradigm only; feed-forward algorithms are, quite simply, barred from participation.

Another important consequence

of the `ntpd`-oriented nature of the existing system clock synchronization architecture is that the feedback loop encourages a separation of the algorithm “intelligence” between user space and the kernel. The `ntpd` daemon operates in the former, but the system clock has its own adaptive procedures that are, in effect, a secondary synchronization system. Two feedback systems linked via a feedback loop are difficult to keep stable, maintain, and even understand. In contrast, by making raw counter values accessible from user space, feed-forward algorithms can avoid the secondary system altogether and concentrate the algorithm intelligence and design in a single, well-defined place. The division of labor is then clean: the synchronization algorithm is in charge of synchronization; the kernel handles the timestamping.

Note that, exactly as in the causal clock $C_c(t)$, a counter can be scaled by a constant so that it reads in more convenient and universal units, and this will not affect the feed-forward algorithm’s ability to synchronize based on it. Linux’s `CLOCK_MONOTONIC_RAW`, for example, provides such a scaled counter, which ticks in (approximate and drifting) nanoseconds.

Currently, the RADclock gets around the lack of feed-forward support by providing patches that extend these mechanisms in FreeBSD and Linux in minimal ways to allow raw counter access to both the kernel and user space. The RADclock API includes difference and absolute clock reading functions based on direct counter timestamping, combined with the latest clock parameters and drift estimate.

The Forgotten Challenge

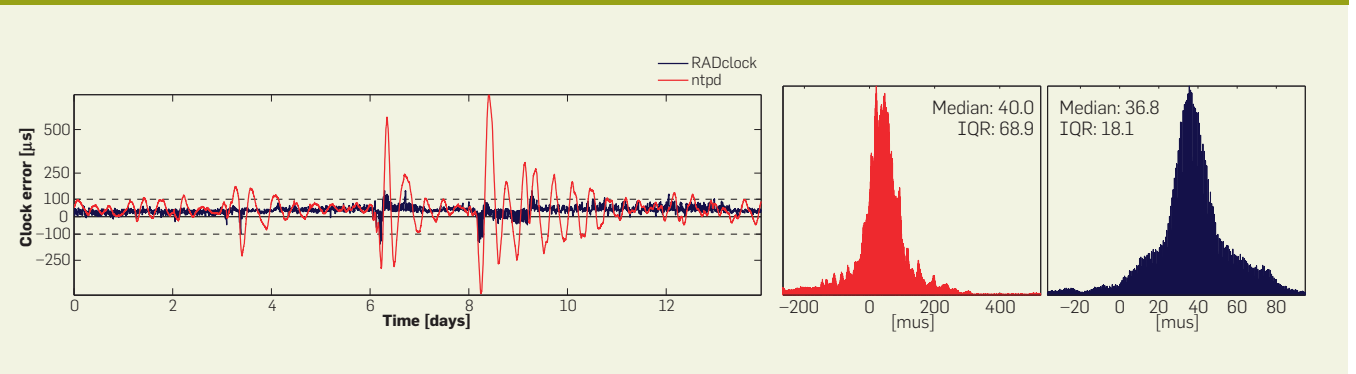
Before continuing, we should point out an annoying truth: *synchronization over networks is actually impossible*. To understand this, let’s reduce the problem to its essence by assuming zero network congestion and system load, so that all delays take their constant, minimal values.

Let $A = d^\uparrow - d^\downarrow$ denote the true path asymmetry, where $d^\uparrow$ and $d^\downarrow$ are the true minimum one-way delays to and from the server, respectively; and let $r = d^\uparrow + d^\downarrow$ be the minimal RTT (see Figure 2). The problem is the existence of a fundamental ambiguity because path asymmetry cannot be measured independently of clock error. The essence of this is the following: if I receive a timestamp T from the server at $t = 1.1$ reading $T = 1$, I cannot tell if I am perfectly synchronized to a server $d^\downarrow = 0.1$ seconds away or, alternatively, if I am 1 second behind true time (packet actually arrived at $t = 2.1$) and the server is 1.1 seconds away.

The asymmetry ambiguity cannot be circumvented, even in principle, by any algorithm. There is, however, some good news. First, this is a problem that plagues absolute clocks only; difference clocks are unaffected. Second, bidirectional exchanges allow constraints flowing from causality (packets can’t arrive before they are sent) to act. Hence, the asymmetry and the associated clock error are boxed in, even if they can’t be known precisely.

The question remains, how are absolute clocks achieving the impossible? What are they returning? In practice, in the absence of any external side-information on A , we must guess a value, and $\hat{A} = 0$ is typically chosen corresponding

Figure 5. Performance of RADclock and ntpd over 14 days synchronizing to a Stratum-1 server on the LAN with a 1024 s polling period; time series (left) and histograms (right). Stability issues are encountered for the feedback control of `ntpd`, resulting in an IQR (Inter-Quartile Range) which is four times wider.



to a symmetric path. This allows the clock to be synchronized, but only up to an unknown error E lying somewhere in the range $[-r, r]$. This range can be tens of milliseconds wide in some cases and can dwarf other errors.

There is another key point to remember here. It is that any change in either the true asymmetry (say because of a routing change), or the estimate of it used by the algorithm, makes the clock jump. For example, based on some knowledge of the routing between the host and the server $r = 15$ ms away, an intrepid administrator may replace the default $\hat{A} = 0$ with a best guess of $\hat{A} = 3$ ms, resulting in a jump of $3/2 = 1.5$ ms. Another example is a change in choice of server, which inevitably brings with it a change in asymmetry. The point is that jumps, even if they result in improvements in synchronization, are an evil unto themselves. Such *asymmetry jitter* can confuse software not only in this host, but also in others, since all OWD's measured to and from the host will also undergo a jump.

To summarize, network synchronization consists of two very different aspects. The synchronization algorithm's role is to see through and eliminate delay variability. It is considered to be accurate if it does this successfully even if the asymmetry error and, therefore the final clock error, is large, as it cannot do anything about this. The asymmetry jitter problem is not about variability but an unknown constant. This is so much simpler; however, it is inherently hard as it cannot be circumvented, even in principle. So here is the challenge: although it cannot be eliminated, *the practical impact of asymmetry depends strongly on how it is managed*. These two very different problems cross paths in a key respect: both benefit from nearby servers.

Robust Algorithm Design

Here is a list of the key elements for reliable synchronization.

Don't forget physics. The foundation of the clock is the local hardware. Any self-respecting algorithm should begin by incorporating the essence of its behavior using a physically meaningful model. Of particular importance are the following simple characterizations of its stability: the large-scale rate error bound (0.1 ppm) and the timescale



With feedforward, the hardware reality is kept in direct view rather than seeing it through algorithmic glasses.



where the oscillator variability is minimal ($\tau = 1,000$ seconds). These characteristics are remarkably stable across PC architectures, but the algorithm should nonetheless be insensitive to their precise values.

The other fundamental physics component is the nature of the delays. A good general model for queuing delays, and hence OWD and RTT, is that of a constant plus a positive random noise. This constant (the minimum value) is clearly seen in Figure 3 in the case of RTT.

Use feed-forward. A feed-forward approach offers far higher robustness, which is essential in a noisy unpredictable environment such as the Internet. It also allows a difference clock to be defined, which in turn is the key to robust filtering for the absolute clock, as well as being directly valuable for the majority of timing applications including network measurement (OWDs aside).

The difference clock comes first. Synchronizing the difference clock equates to measuring the long-term average period p_{av} . This “rate synchronization” is more fundamental, more important, and far easier than absolute synchronization. A robust solution for this is a firm foundation on which to build the much trickier absolute synchronization.

Note that by *rate synchronization* we mean a low noise estimate of *average long-term rate/period*, not to be confused with short-term rate, which reduces essentially to (the derivative of) drift, and hence to absolute synchronization.

Use minimum RTT-based filtering. If a timing packet is lucky enough to experience the minimum delay, then its timestamps have not been corrupted and can be used to set the absolute clock directly to the right value (asymmetry aside). The problem is, how can we determine which packets get lucky?

Unfortunately, this is a chicken-and-egg problem, since to measure OWD, we have to use the absolute clock $C_a(t)$, which is the one we want to synchronize in the first place. Luckily, the situation is different with RTTs, which can be measured by the difference clock $C_d(t)$. The key consequence is that a reliable measure of packet quality can be obtained without the need to first absolutely synchronize the clock. One just measures by how much the RTT

of the given timing packet exceeds the minimum RTT: the smaller the gap, the higher the ‘quality’ of the packet.

The sharp-eyed reader would have noticed that a chicken-and-egg problem remains. The difference clock also needs filtering in order to synchronize it, so how can we use it before it is synchronized? The answer is that even a very poor estimate of p_{av} is sufficient to perform meaningful filtering. For example, imagine that p_{av} is known to an accuracy of 100 ppm (which is truly atrocious) and that $RTT=10ms$. Then the error in the RTT measurement (including that resulting from drift) is of the order of $1\ \mu s$, which is well below the noise caused by the operating system (around $10\ \mu s$).

Rate synchronization. Let’s assume that our filtering is successful, so we can reliably measure the level of quality of timing packets.

The key to rate synchronization (that is, the measurement of p_{av}) is that it is a long-term average, which means that we can afford to be patient! Reliable rate synchronization is then as simple as collecting reasonable-quality timestamps at either end of a large interval (ideally several days, but even much less will work well). Timestamp errors and errors in the quality assessment are then compressed by the size of the interval and enter into the 0.001-ppm zone where they will never bother you again.

Once synchronized, the p_{av} value has a long life. One consequence is that even if connectivity to a server were lost, the difference clock would remain well synchronized. Using a

hardware-validated test bed, we performed tests showing that the RAD-clock difference clock measured one-second intervals to sub- μs accuracy even after 20 days of disconnection from the server. Compare this level of robustness with that of absolute synchronization, where over such an interval the local clock will inevitably drift considerably, or worse.

Absolute synchronization. Drift is constantly happening, and a clock must be ready to be read at any time. It follows that patience is not an option here: even if congestion is high and timing packets delayed, the best must still be made of a bad situation; the drift must be dealt with.

There are two chief considerations. The first is to use the timing packet quality to control its contribution to the clock error estimation. The control must be very strict: if the packet has an RTT that is only a little above the minimum, that gap corresponds directly to an error to be avoided.

The second consideration is time scale. The essential trade-off is the need to have data from as many timing packets as possible to increase the chances of getting lucky, versus the need for them to be recent so that the current error, rather than an outdated error from the past, is estimated. Here the time scale τ plays a key role, as it places a meaningful limit on how far back in the past to look.

The absolute clock can be defined off the difference clock simply as follows:

$$C_a(t) = C_d(t) - \hat{E}(t)$$

where $\hat{E}(t)$ is the estimate of the error of $C_d(t)$, which is removed on the fly when the absolute clock is read by a process. Figure 6 shows $\hat{E}(t)$ as the blue curve, which is calculated based on the noisy unfiltered per-packet error estimates that underlie it (green spikes). The true error $E(t)$ is the black curve, as measured using an external GPS-synchronized DAG capture card¹ in our test bed.

One server is enough. So far we have discussed synchronizing to a single server. Surely one could connect to multiple servers and select among them to obtain even better results? In principle this is true; in Internet practice, at least with the current state of the art, it is most definitely not, for two reasons.

Most fundamentally, switching servers implies a change in path asymmetry and hence a jump in the clock. Imagine a server-selection algorithm that is moving around among candidate servers of similar quality. The result is constant jumping—a classic case of asymmetry jitter. Such jitter is not hard to observe in `ntpd`, where a connection to multiple peers is in fact recommended.

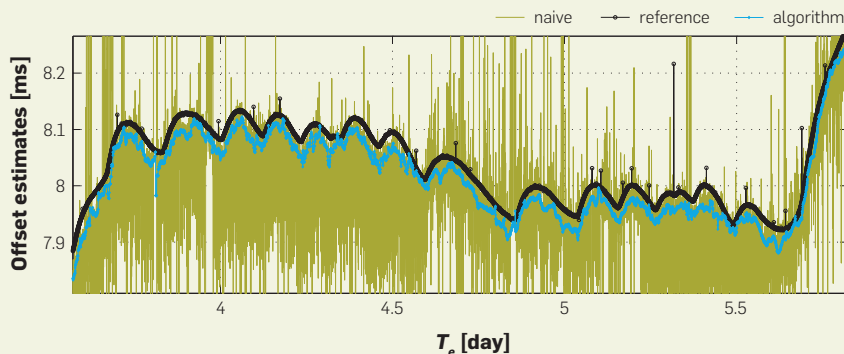
Second, once performance is tuned to close to system and network noise limits, any disruption at all will downgrade it. Ceasing to query a server for a time and then returning to it, for example, qualifies as a moderate to large disruption.

The take-home message for administrators is this: if you want to improve your system clock performance, just make sure the configuration points only to a single (nearby) server.

Watch your route. Servers do change, and even if they don’t, the routing to them will eventually. Not only will this result in an unavoidable jump for asymmetry reasons, it is also a challenge to the synchronization algorithm itself. Careful design is needed so that the rate and absolute synchronization algorithms transition gracefully and robustly to the new environment. This has to begin with a well-thought-out approach to the quality assessment underlying the filtering. If this breaks down, then quality packets can be assessed as good and good as bad, and there is no limit to the potential damage.

Trust yourself. Yes, the server is the expert, without which synchronization would be impossible. Still, it should

Figure 6. An unsynchronized clock drifting. The black line is the drift, the green the drift seen through the noise of delay variability, and the blue the estimate of the drift by the RADclock algorithm (it is offset vertically from the black by about $30\ \mu s$ because path asymmetry has not been corrected for here). Source: Veitch et al.⁴



not be trusted. Servers can and do have their bad periods, and blind faith in them can lead to deep trouble. Fortunately another authority is available: the counter model. If the RTT filtering is telling you that congestion is low, yet the server timestamps are saying you are suddenly way out, trust the hardware and use the model to sanity-check the server. Basically, a drift well over 1 ppm is not credible, and the algorithm should smell a rat.

When in doubt, just drift. What if congestion has become so high that none of the available timestamps is of acceptable quality? What if I don't trust the server, or if I have lost contact with it entirely?

There is only one thing to do: sit back and relax. Nothing bad will happen unless the algorithm chooses to make it happen. A reaction of inaction is trivial to implement within the feed-forward paradigm and results in simply allowing the counter to drift gracefully. Remember the counter is highly stable, accumulating only around 1 μ s per second, at worst.

More generally, the algorithm should be designed never to overreact to anything. Remember, its view of the world is always approximate and may be wrong, so why try to be too clever when inaction works so well? Unfortunately, feedback algorithms such as `ntpd` have more reactive strategies that drive the clock more strongly in the direction of their opinions. This is a major source of their nonrobustness to disruptive events.

The Bigger Picture

Thus far we have considered the synchronization of a single host over the network to a server, but what about the system as a whole? In NTP, the main system aspect is the server hierarchy. In a nutshell, Stratum-1 servers anchor the tree as they use additional hardware (for example, a PC with a GPS receiver or a purpose-built synchronization box with GPS) to synchronize locally, rather than over a network. By definition, Stratum-2 servers synchronize to Stratum-1, Stratum-3 to Stratum-2, and so on, and hosts synchronize to whatever they can find (typically a Stratum-2 or a public Stratum-1).

At the system level there are a number of important and outstanding chal-



Yes, the server is the expert, without which synchronization would be impossible. Still, it should not be trusted. Servers can and do have their bad periods, and blind faith in them can lead to deep trouble.



lenges. Stratum-1 servers do not communicate among themselves, but act (except for load balancing in limited cases) as independent islands. There is a limited capability to query a server individually to obtain basic information such as whether it is connected to its hardware and believes it is synchronized, and there is no ability to query the set of servers as a whole. An interconnected and asymmetry-aware Stratum-1 infrastructure could provide a number of valuable services to clients. These include recommendations about the most appropriate server for a client, automatic provision of backup servers taking asymmetry jitter into account, and validated information on server quality. Currently no one is in a position to point the finger at flaky servers, but they are out there.

Building on the RADclock algorithm, the RADclock project³ aims to address these issues and others as part of a push to provide a robust new system for network timing within two years. Details for downloading the existing client and server software (packages for FreeBSD and Linux), documentation, and publications can be found on the RADclock project page.

Acknowledgments

The RADclock project is partially supported under Australian Research Council's Discovery Projects funding scheme (project number DP0985673), the Cisco University Research Program Fund at Silicon Valley Community Foundation, and a Google Research Award. 

References

1. Endace Measurement Systems. DAG series PCI and PCI-X cards; <http://www.endace.com/networkMCards.htm>.
2. Mills, D.L. 2006. *Computer Network Time Synchronization: The Network Time Protocol*. CRC Press, Boca Raton, FL, 2006.
3. Ridoux, J., Veitch, D. RADclock Project Web page; <http://www.cubinlab.ee.unimelb.edu.au/radclock/>.
4. Veitch, D., Ridoux, J., Korada, S.B. Robust synchronization of absolute and difference clocks over networks. *IEEE/ACM Transactions on Networking* 17, 2 (2009), 417–430. doi: 10.1109/TNET.2008.926505.

Julien Ridoux is a Research Fellow at the Center for Ultra-Broadband Information Networks (CUBIN) in the Department of Electrical & Electronic Engineering at The University of Melbourne, Australia.

Darryl Veitch is a Principal Research Fellow at the Center for Ultra-Broadband Information Networks (CUBIN) in the Department of Electrical & Electronic Engineering at The University of Melbourne, Australia.

The competition among cloud providers may drive prices downward, but at what cost?

BY DAVE DURKEE

Why Cloud Computing Will Never Be Free

THE LAST TIME the IT industry delivered outsourced shared-resource computing to the enterprise was with timesharing in the 1980s when it evolved to a high art, delivering the reliability, performance, and service the enterprise demanded. Today, cloud computing is poised to address the needs of the same market, based on a revolution of new technologies, significant unused computing capacity in corporate data centers, and the development of a highly capable Internet data communications infrastructure. The economies of scale of delivering computing from a centralized, shared infrastructure have set the expectation among customers that cloud computing costs will be significantly lower than those incurred from providing their own computing. Together with the reduced deployment costs of open source software and the

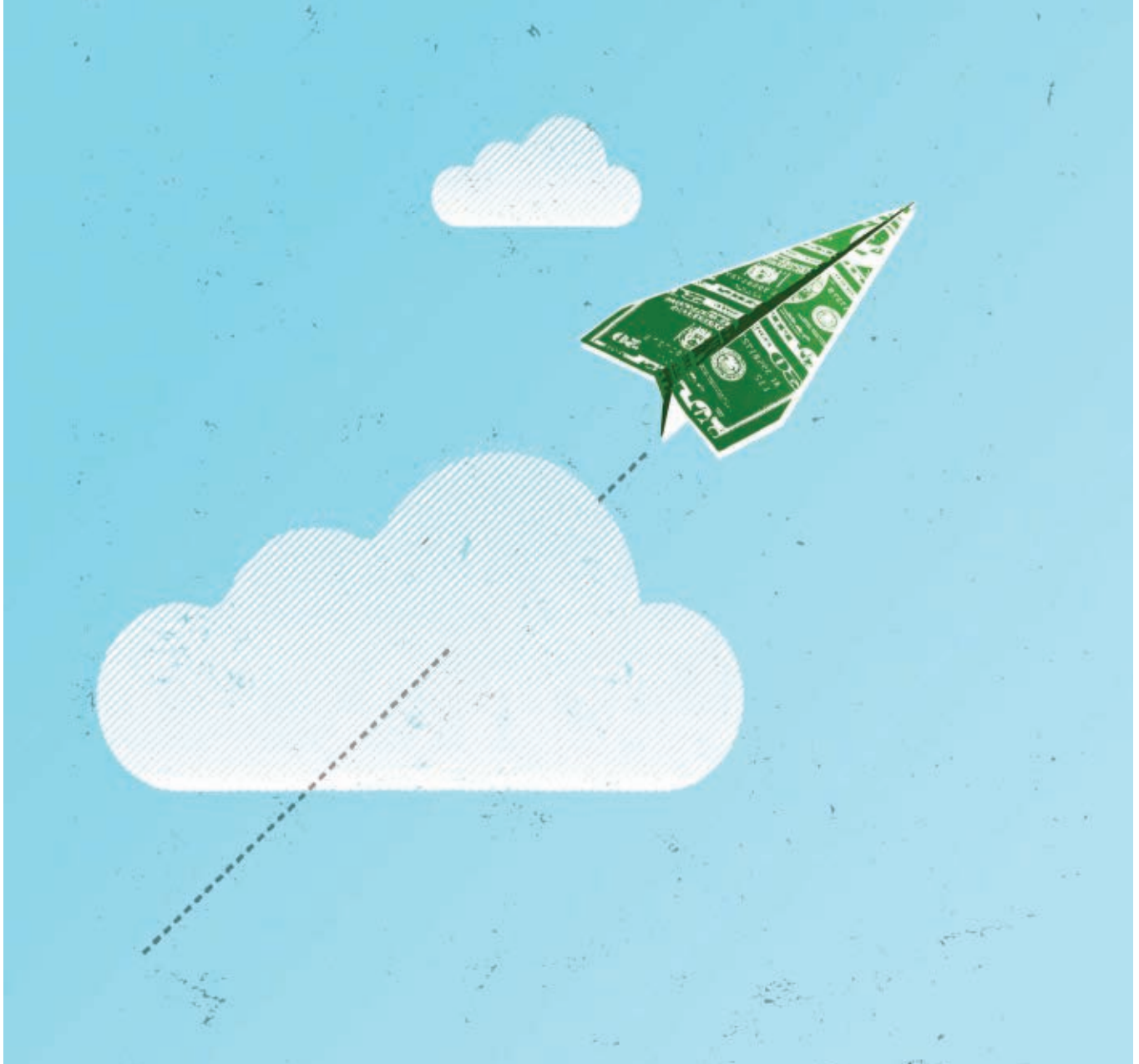
perfect competition characteristics of remote computing, these expectations set the stage for fierce pressure on cloud providers to continuously lower prices.

This pricing pressure results in a commoditization of cloud services that deemphasizes enterprise requirements such as guaranteed levels of performance, uptime, and vendor responsiveness, much as has been the case with the Web hosting industry. Notwithstanding, it is the expectation of enterprise management that operating expenses be reduced through the use of cloud computing to replace new and existing IT infrastructure. This difference between expectation and what the industry can deliver at today's near-zero price points represents a challenge, both technical and organizational, which will have to be overcome to ensure large-scale adoption of cloud computing by the enterprise.

The Essential Characteristics of Cloud Computing

This is where we come full circle and timesharing is reborn. The same forces are at work that made timesharing a viable option 30 years ago: the high cost of computing (far exceeding the cost of the physical systems), and the highly specialized labor needed to keep it running well. The essential characteristics of cloud computing that address these needs are:⁴

- ▶ *On-demand access.* Rapid fulfillment of demand for computing and continuing ability to fulfill that demand as required.
- ▶ *Elasticity.* Computing is provided in the amount required and disposed of when no longer needed.
- ▶ *Pay-per-use.* Much like a utility, cloud resource charges are based on the quantity used.
- ▶ *Connectivity.* All of the servers are connected to a high-speed network that allows data to flow to the Internet as well as between computing and storage elements.
- ▶ *Resource pooling.* The cloud provider's infrastructure is shared across



some number of end customers, providing economies of scale at the computing and services layers

- *Abstracted infrastructure.* The cloud end customer does not know the exact location or the type of computer(s) their applications are running on. Instead, the cloud provider provides performance metrics to guarantee a minimum performance level.

- *Little or no commitment.* This is an important aspect of today's cloud computing offerings, but as we will see here, interferes with delivery of the services the enterprise demands.

Service Models

Cloud is divided into three basic service models. Each model addresses a specific business need.

Infrastructure as a Service (IaaS). This is the most basic of the cloud

service models. The end customer is purchasing raw compute, storage, and network transfer. Offerings of this type are delivered as an operating system on a server with some amount of storage and network transfer. These offerings can be delivered as a single server or as part of a collection of servers integrated into a virtual private data center (VPDC).

Platform as a Service (PaaS). This is the next layer up where the end customer is purchasing an application environment on top of the bare bones infrastructure. Examples of this would be application stacks: Ruby on Rails, Java, or LAMP. The advantage of PaaS is that the developer can buy a fully functional development and/or production environment.

Software as a Service (SaaS). This currently is the highest layer in the cloud

stack. The end customer is purchasing the use of a working application. Some examples of this are NetSuite and Salesforce.com. (This service is not the focus of this article.)

Perfect Competition Determines Cloud Pricing Strategies

In our experience providing cloud services, many of the current cloud end customers use price as their primary decision criteria. As a result, service providers' offerings tend toward a least common denominator, determined by the realities of providing cloud service at the lowest possible price. At the same time, the cloud computing market is becoming more crowded with large providers entering the playing field, each one of which trying to differentiate itself from the already established players. The result of many pro-

viders competing to deliver very similar product in a highly price-competitive environment is termed *perfect competition* by economists. Perfectly competitive markets, such as those for milk, gasoline, airline seats, and cellphone service, are characterized by a number of supplier behaviors aimed at avoiding the downsides of perfect competition, including:

- ▶ Artificially differentiating the product through advertising rather than unique product characteristics
- ▶ Obscuring pricing through the use of additional or hidden fees and complex pricing methodologies
- ▶ Controlling information about the product through obfuscation of its specifications
- ▶ Compromising product quality in an effort to increase profits by cutting corners in the value delivery system
- ▶ Locking customers into long-term commitments, without delivering obvious benefits.

These factors, when applied to the cloud computing market, result in a product that does not meet the enterprise requirements for deterministic behavior and predictable pricing. The resulting price war potentially threatens the long-term viability of the cloud vendors. Let's take a closer look at how perfect competition affects the cloud computing market.

Variable performance. We frequently see advertisements for cloud computing breaking through the previous price floor for a virtual server instance. It makes one wonder how cloud providers can do this and stay in business. The answer is that they over commit their computing resources and cut corners on infrastructure. The result is variable and unpredictable performance of the virtual infrastructure.⁵

Many cloud providers are vague on the specifics of the underlying hardware and software stack they use to deliver a virtual server to the end customer, which allows for overcommitment. Techniques for overcommitting hardware include (but are not limited to):

- ▶ Specify memory allocation and leave CPU allocation unspecified, allowing total hardware memory to dictate the number of customers the hardware can support;
- ▶ Quote shared resource maximums instead of private allocations;



The cloud computing market is becoming more crowded with large providers entering the playing field, each one of which trying to differentiate itself from the already established players.



▶ Offer a range of performance for a particular instance, such as a range of GHz; and

▶ Overallocate resources on a physical server, or “thin provisioning.” Commercial virtualization management software such as VMWare or Virtuozzo offer the ability to overallocate resources on the underlying hardware, resulting in reduced performance during peak loads.

Like overcommitment, limiting access to infrastructure resources or choosing lower-priced, lower-performance (and potentially older) infrastructure is used by vendors to make providing cloud computing at rock-bottom prices viable. We entered the cloud provider business after discovering we could not guarantee enterprise-grade performance to our customers by reselling other vendors' cloud services due to their corner-cutting. Here is a list of some of the strategies the author has seen over the years:

▶ *Traffic shaping.* A new client asked us to move their existing VPDC to our infrastructure. Shortly after initiating the transfer, the data rate dropped from approximately 10Mbit/sec to 1Mbit, where it remained for the duration of the transfer. This behavior speaks pretty strongly of traffic shaping. Because the client based their downtime window for the data center move on the assumption that the connecting network was gigabit Ethernet, they missed their estimate by over a week.

▶ *Using older gigabit or fast Ethernet networking.* An ISP that was selling an Amazon-like cloud computing product connected their servers to the Internet using fast Ethernet switches. If a customer wanted faster connectivity, there was an up charge per port.

▶ *Recycling failed disk drives.* A client leased several servers in a private cloud from a very large ISP. He had previously experienced several drive failures with this ISP, so he decided to check up on the hardware by running `smartctl` to assess the health of the drives. What he found was shocking to him: the ‘new’ servers he had just received had disk drives in them that were over three years old! When he challenged the ISP, he was told their policy was to replace a drive only when it fails.

▶ *Deploying older CPU technology.* We were asked to manage a client's ap-

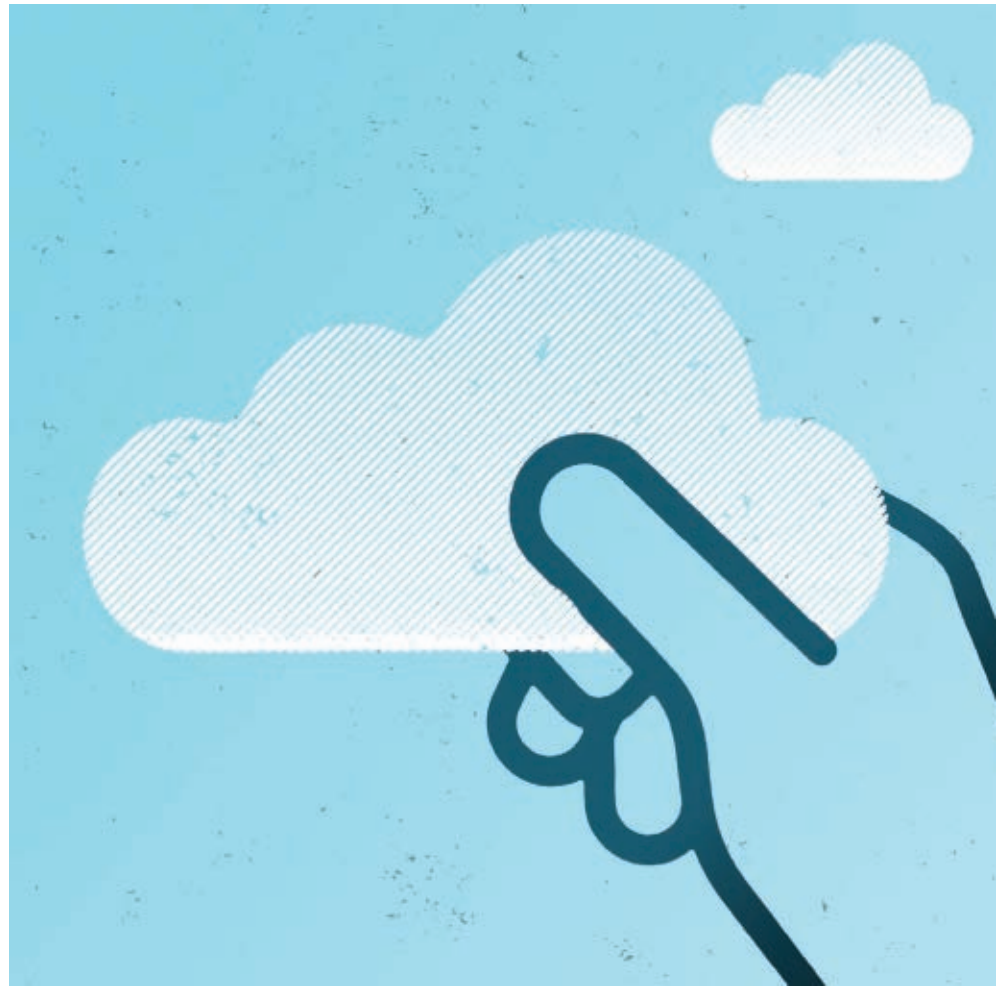
plications that were hosted at another ISP. As part of our initial assessment of the client's environment, we discovered that he had received Athlon desktop processors in the servers that he was paying top dollar for.

This difference between advertised and provided value is possible because cloud computing delivers abstracted hardware that relieves the client of the responsibility for managing the hardware, offering an opportunity for situations such as those listed here to occur. As our experience in the marketplace shows, the customer base is inexperienced with purchasing this commodity and overwhelmed with the complexity of selecting and determining the cost of the service, as well being hamstrung by the lack of accurate benchmarking and reporting tools. Customer emphasis on pricing levels over results drives selection of poorly performing cloud products. However, the enterprise will not be satisfied with this state of affairs.

Extra charges. For example, ingress and egress bandwidth is often charged separately and using different rates, overages on included baseline storage, or bandwidth quantities are charged at much higher prices than the advertised base rates, charges are applied to the number of IOPS used on the storage system, and charges are levied on HTTP get/put/post/list operations, to name but a few. These additional charges cannot be predicted by the end user when evaluating the service, and are another way the cloud providers are able to make the necessary money to keep their businesses growing because the prices they are charging for compute aren't able to support the costs of providing the service. The price of the raw compute has become a loss-leader.

Long-term commitments. Commitment hasn't been a prominent feature of cloud customer-vendor relationships so far, even to the point that pundits will tell you that "no commitment" is an essential part of the definition of cloud computing. However, the economics of providing cloud computing at low margins is changing the landscape. For example, Amazon AWS introduced reserved instances that require a one- or three-year long commitment.

There are other industries that offer their services with a nearly identical



delivery model, most obviously cellular telephone providers and to some extent electrical utilities. However, for some reason, cloud computing is not delivered with the same pricing models as those developed over the last hundred years to deliver electricity. These providers all use long-term commitments to ensure their economic viability by matching their pricing to customer resource usage that determines their costs. Long-term commitments—in other words, contracts—allow for time-of-use pricing and quantity discounts. We feel these characteristics will become ubiquitous features of cloud computing in the near future. For cloud computing delivered as SaaS, long-term commitments are already prevalent.

Navigating Today's Perfectly Competitive Cloud Computing Market

Today's price-focused cloud computing market, which is moving rapidly toward perfect competition, presents

challenges to the end customer in purchasing services that will meet their needs. This first-generation cloud offering, essentially Cloud 1.0, requires the end customer to understand the trade-offs that the service provider has made in order to offer computing to them at such a low price.

Service-Level Agreements. Cloud computing service providers typically define an SLA as some guarantee of how much of the time the server, platform, or application will be available. In the cloud market space, meaningful SLAs are few and far between, and even when a vendor does have one, most of the time it is toothless. For example, a well-known cloud provider guarantees an availability level of 99.999% uptime, or five minutes a year, with a 10% discount on their charges for any month in which it is not achieved. However, since their infrastructure is not designed to reach five-nines of uptime, they are effectively offering a 10% discount on their services in exchange for the benefit of claiming that level of reli-



ability. If a customer really needs five-nines of uptime, a 10% discount is not going to even come close to the cost of lost revenue, breach of end-user service levels, or loss of market share due to credibility issues.

Another trick the service providers play on their customers is to compute the SLA on an annualized basis. This means that customers are only eligible for a service credit after one year has passed. Clearly the end user should pay close attention to the details of the SLA being provided and weigh that against what business impact it will have if the service provider misses the committed SLA. From what we have seen in the last four years of providing IaaS and PaaS, most customers do not have a strong understanding of how much downtime their businesses can tolerate or what the costs are for such downtime. This creates a carnival atmosphere in the cloud community where ever higher SLAs are offered at lower prices without the due diligence needed to achieve them...another race to the bottom.

Taking advantage of the low prices of Cloud 1.0 requires an honest assessment by the end customer of the level of reliability they actually need.

Performance is almost never discussed. One of the hazards of shared infrastructure is that one customer's usage patterns may affect other customers' performance. While this interference between customers can be engineered out of the system, addressing this problem is an expense that vendors must balance against the selling price. As a result, repeatable benchmarks of cloud performance are few and far between because they are not easily achieved, and Cloud 1.0 infrastructure is rarely capable of performance levels that the enterprise is accustomed to.

While it makes intuitive sense to quiz the cloud provider on the design of their infrastructure, the universe of possibilities for constraining performance to achieve a \$0.03/hour instance price defies easy analysis, even for the hardware-savvy consumer. At

best, it makes sense to ask about performance SLAs, though at this time we have not seen any in the industry. In most cases, the only way to determine if the service meets a specific application need is to deploy and run it in production, which is prohibitively expensive for most organizations.

In my experience, most customers use CPU-hour pricing as their primary driver during the decision-making process. While the resulting performance is adequate for many applications, we have also seen many enterprise-grade applications that failed to operate acceptably on Cloud 1.0.

Service and support. One of the great attractions of cloud computing is that it democratizes access to production computing by making it available to a much larger segment of the business community. In addition, the elimination of the responsibility for physical hardware removes the need for data center administration staff. As a result, there is an ever-increasing number of people responsible for production computing who do not have system administration backgrounds, which creates demand for comprehensive cloud vendor support offerings. Round-the-clock live support staff costs a great deal and commodity cloud pricing models cannot support that cost. Many commodity cloud offerings have only email or Web-based support, or only support the usage of their service, rather than the end-customer's needs.

When you can't reach your server just before that important demo for the new client, what do you do? Because of the mismatch between the support levels needed by cloud customers and those delivered by Cloud 1.0 vendors, we have seen many customers who replaced internal IT with cloud, firing their system administrators, only to hire cloud administrators shortly thereafter. Commercial enterprises running production applications need the rapid response of phone support delivered under guaranteed SLAs.

Before making the jump to Cloud 1.0, it is appropriate to consider the costs involved in supporting its deployment in your business.

The Advent of Cloud 2.0: The Value-Based Cloud

The current myopic focus on price has

created a cloud computing product that has left a lot on the table for the customer seeking enterprise-grade results. While many business problems can be adequately addressed by Cloud 1.0, there are a large number of business applications running in purpose-built data centers today for which a price-focused infrastructure and delivery model will not suffice. For that reason, we see the necessity for a new cloud service offering focused on providing value to the SME and large enterprise markets. This second-generation value-based cloud is focused on delivering a high performance, highly available, and secure computing infrastructure for business-critical production applications, much like the mission of today's corporate IT departments.

This new model will be specifically designed to meet or exceed enterprise expectations, based on the knowledge that the true cost to the enterprise is not measured by the cost per CPU cycle alone. The reasons most often given by industry surveys of CIOs for holding back on adopting the current public cloud offerings are that they do not address complex production application requirements such as compliance, regulatory, and/or compatibility issues. To address these issues, the value-based cloud will be focused on providing solutions rather than just compute cycles.

Cloud 2.0 will not offer CPU at \$0.04/hour. Mission-critical enterprise applications carry with them a high cost of downtime.² Indeed, many SaaS vendors offer expensive guarantees to their customers for downtime. As a result, enterprises typically require four-nines (52 minutes unavailable a year) or more of uptime. Highly available computing is expensive, and historically, each additional nine of availability doubles the cost to deliver that service. This is because infrastructure built to provide five-nines of availability has no single points of failure and is always deployed in more than one physical location. Current cloud deployment technologies use n+1 redundancy to improve on these economies up to the three-nines mark, but they still rule past this point. Because the cost of reliability goes up geometrically as the 100% mark is neared, many consider five-nines and above to be nearly unachievable (and unaffordable), only



The current myopic focus on price has created a cloud computing product that has left a lot on the table for the customer seeking enterprise-grade results.



deserving of the most mission-critical applications. In addition, there are significant infrastructure challenges to meet the performance requirements of the enterprise, which significantly raise resource prices.

Technology challenges faced by Cloud 2.0 providers. The number-one problem that Cloud 2.0 providers face is supplying their enterprise customers with storage that can match the performance and reliability they are accustomed to from their purpose-built data centers at a price point that is significantly lower. When traditional storage technologies are used in a cloud infrastructure, they fail to deliver adequate performance because the workload is considerably less predictable than what they were designed for. In particular, the randomness of disk accesses as well as the working set size are both proportional to the number of different applications that the storage system is serving at once. Traditionally SANs have solved the problem of disk read caching by using RAM caches. However, in a cloud application, the designed maximum RAM cache sizes are completely inadequate to meet the requirement of caching the total working sets of all customer applications. This problem is compounded on the write side, where the caches have traditionally been battery-backed RAM, which is causing storage vendors to move to SSD technology to support cloud applications.

Once the storage caching problem has been solved, the next issue is getting cloud applications' large volumes of data out of the SAN into the server. Legacy interconnect, such as fiber-channel with which most SANs are currently shipped, cannot meet the needs of data-hungry Cloud 2.0 infrastructures. Both Ethernet and Infiniband offer improved performance, with currently shipping Infiniband technology holding the title of fastest available interconnect. Storage vendors who eschew Infiniband are relegating their products to second-tier status in the Cloud 2.0 world. Additionally, fast interconnect is a virtual requirement between servers, since enterprise applications are typically deployed as virtual networks of collaborating instances that cannot be guaranteed to be on the same physical servers.¹

With an increasing number of clouds being deployed in private data centers or small-to-medium MSPs, the approach to build a cloud used by Amazon, in which hardware and software were all developed in-house, is no longer practical. Instead, clouds are being built out of commercial technology stacks with the aim of enabling the cloud vendor to go to market rapidly while providing high-quality service. However, finding component technologies that are cost competitive while offering reliability, 24x7 support, adequate quality (especially in software), and easy integration is extremely difficult, given that most legacy technologies were not built or priced for cloud deployment. As a result, we expect some spectacular Cloud 2.0 technology failures, as was the case with Cloud 1.0. Another issue with this approach is that the technology stack must provide native reliability in a cloud configuration that actually provides the reliability advertised by the cloud vendor.

Why transparency is important. Transparency is one of the first steps to developing trust in a relationship. As we discussed earlier, the price-focused cloud has obscured the details of its operation behind its pricing model. With Cloud 2.0, this cannot be the case. The end customer must have a quantitative model of the cloud's behavior. The cloud provider must provide details, under NDA if necessary, of the inner workings of their cloud architecture as part of developing a closer relationship with the customer. Insight into the cloud provider's roadmap and objectives also brings the customer into the process of evolving the cloud infrastructure of the provider. Transparency allows the customer to gain a level of trust as to the expected performance of the infrastructure and the vendor. Taking this step may also be necessary for the vendor to meet enterprise compliance, and/or regulatory requirements.

This transparency can only be achieved if the billing models for Cloud 2.0 clearly communicate the value (and hence avoided costs) of using the service. To achieve such clarity, the cloud vendor has to be able to measure the true cost of computing operations that the customer executes and bill for them. Yet today's hardware, as well as management, monitoring, and billing



Transparency allows the customer to gain a level of trust as to the expected performance of the infrastructure and the vendor. Taking this step may also be necessary for the vendor to meet enterprise compliance, and/or regulatory requirements.



software are not designed to provide this information. For example, billing for IOPs in a multitenant environment is a very deep technological problem, impacting not only the design of the cloud service, but the technologies it rests on such as operating systems, device drivers, and network infrastructure. Another example is computing and minimizing the costs of fragmentation of computing resources across one or more clusters of compute nodes while taking into consideration the time dependence of individual customers' loads and resource requirements.

The role of services. When cloud infrastructure reduces the barriers to deployment, what still stands in the way? That would be services, such as ongoing administration, incident response, SLA assurance, software updates, security hardening, and performance tuning. Since 80% of downtime is caused by factors other than hardware,³ services are essential to reliable production computing. Traditionally these services have been delivered by the enterprise's IT department, and simply replacing their servers with remote servers in the cloud doesn't solve the services problem. Because services delivered with cloud computing will necessarily be outsourced, they must be delivered within the context of a long-term commitment that allows the vendor to become familiar with the customer's needs, which will retire today's Cloud 1.0 customer expectation of little or no commitment. At the same time, the move toward long-term commitments will drive vendors to focus on customer satisfaction rather than the more prevalent churn visible in perfectly competitive markets.

Service-level management. Service-level agreements are the name of the game in Cloud 2.0. Enterprise customers typically have obligations to provide services to their customers within a contracted SLA. The service delivery infrastructure's SLA must meet or exceed the service levels that the enterprise has committed to provide. All aspects of the service delivery infrastructure (compute fabric, storage fabric, and network fabric) should be monitored by a monitoring system. In addition, all of the customer's cloud instances should be monitored as well. VMs must be monitored at the

system level as well as the application level. The monitoring system's rich data collection mechanisms are then fed as inputs to the service providers' processes so that they can manage service-level compliance. A rich reporting capability to define and present the SLA compliance data is essential for enterprise customers. Typically, SLAs comprise of some number of service-level objectives (SLOs). These SLOs are then rolled up to compute the overall SLA. It pays to remember the overall SLA depends on the entire value delivery system, from the vendor's hardware and software to the SLOs for the vendor's support and operations services offerings. To provide real value to the enterprise customer, the cloud provider must negotiate with the customer to deliver their services at the appropriate level of abstraction to meet the customer's needs, and then manage those services to an overall application SLA.

The role of automation. In order to obtain high quality and minimize costs the value-based cloud must rely on a high degree of automation. During the early days of SaaS clouds, when the author was building the NetSuite data center, they had over 750 physical servers that were divided into three major functions: Web delivery, business logic, and database. Machine-image templates were used to create each of the servers in each tier. However, as time went on, the systems would diverge from the template image because of ad hoc updates and fixes. Then, during a deployment window, updates would be applied to the production site, often causing it to break, which resulted in a violation of end-customer SLAs. As a consequence, extensive effort was applied to finding random causes for the failed updates. The root cause was that the QA tests were run on servers that were exact copies of the templates; however, some of the production systems were unique, which caused faults during the deployment window. These types of issues can break even the tightest of deployment processes. The moral of the story is to never log in to the boxes. This can only be accomplished by automating all routine system administration activities.

There are several data-center-run book-automation tools on the market

today for use in corporate data centers. These tools allow for the complete automation of every aspect of the server life cycle from creation of a virtual infrastructure through scaling, service-level management, and disposal of the systems when the customer has finished with them. While automation has made significant progress in the corporate data center, it is only in its infancy in the cloud. Yet, to replace the corporate data center, Cloud 2.0 must include automation. This capability allows both the cloud provider and the customer to obtain some unprecedented benefits:

- ▶ *Very high service levels.* The system is managing itself, and humans get involved only as needed, both at the service provider and customer-level processes.

- ▶ *Problems and solutions become methodological rather than random.* This allows you to fix all instances of a problem with a code change.

- ▶ *Automatically scalable infrastructure.* Allows customers to pay for only what they need when they need it without additional system administration effort to maintain service levels.

- ▶ *Automatic disaster recovery.* Automation handles the manual tasks of failover to a backup data center as well as failing back to the primary data center.

- ▶ *Minimize staffing.* The automation framework uses feedback from the monitoring system to automatically address common solutions to common problems, as well as automatically execute repetitive processes. Escalation to staff occurs only when the automation framework can't address a fault.

- ▶ *Power savings.* The automation framework concentrates the workloads onto the minimum number of servers necessary to maintain service levels, and turns off the rest.

Betting your Business on Cloud 2.0

By offering value beyond simply providing CPU cycles, the cloud provider is becoming a part of the end customers' business. This requires a level of trust that is commensurate with hiring an employee or outsourcing your operations. Do you know who you are hiring? This vendor-partner must understand what the enterprise holds important, and must be able to oper-

ate in a way that will support the cloud end customer's business. By taking on the role of operations services provider to the enterprise, the vendor enables the end customer to gain all of the benefits of cloud computing without the specialized skills needed to run a production data center. However, it is unrealistic to expect outsourced IT that eliminates the need for in-house staffing to be delivered at today's cloud computing prices.

For the Cloud 2.0 revolution to take hold, two transformations must occur, which we are already seeing in our sales and marketing activities: cloud vendors must prepare themselves to provide value to the enterprise that entices them out of their purpose-built data centers and proprietary IT departments; and customers must perceive and demand from cloud vendors the combination of fast and reliable cloud computing with operations services that their end users require. ■

Related articles on queue.acm.org

CTO Roundtable: Cloud Computing

Mache Creeger

<http://queue.acm.org/detail.cfm?id=1536633>

Building Scalable Web Services

Tom Killalea

<http://queue.acm.org/detail.cfm?id=1466447>

Describing the Elephant: The Different Faces of IT as Service

Ian Foster and Steven Tuecke

<http://queue.acm.org/detail.cfm?id=1080874>

References

1. *EE Times Asia*. Sun grooms Infiniband for Ethernet face-off; http://www.eetasia.com/ART_8800504679_590626_NT_e979f375.HTM
2. Hiles, A. Five nines: chasing the dream?; <http://www.continuitycentral.com/feature0267.htm>
3. Jayaswal, K. *Administering Data Centers: Servers, Storage, and Voice Over IP*. John Wiley & Sons, Chicago, IL, 2005; http://searchdatamanagement.techtarget.com/generic/0,295582,sid91_gcl1150917,00.html
4. National Institute of Science and Technology. NIST Definition of Cloud Computing; <http://csrc.nist.gov/groups/SNS/cloud-computing/>
5. Winterford, B. Stress tests rain on Amazon's cloud. *IT News*; <http://www.itnews.com.au/News/153451,stress-tests-rain-on-amazons-cloud.aspx>

Dave Durkee (dave@enikiconsulting.net) is founder and technical director of ENKI, a managed cloud computing services provider in Mountain View, CA. Durkee has over 25 years of experience in IT infrastructure, networking, business applications development, and executive corporate management. He has held several senior management IT positions, including CIO of a successful hosted ERP application service provider, Netsuite.com

© 2010 ACM 0001-0782/10/0500 \$10.00

DOI:10.1145/1735223.1735243

Rather than try to capture everything, system design should focus on the psychological basis of human memory.

BY ABIGAIL SELLEN AND STEVE WHITTAKER

Beyond Total Capture: A Constructive Critique of Lifelogging

WHAT IF WE could digitally capture *everything* we do and see? What if we could save every bit of information we touch and record every event we experience? What would such a personal digital archive be like, and how might it affect the way we live? This vision of a complete “lifelog” is the holy grail for many technologists and researchers who consider us to be on the brink of an “e-memory” revolution.

In the past few years, capturing “Memories for Life” has become a U.K. Grand Challenge in Computing (http://www.nesc.ac.uk/esi/events/Grand_Challenges/proposals/), and many research programs today are dedicated to developing technologies to support the archiving of vast amounts of personal data. A 2009

book² by Gordon Bell and Jim Gemmell outlined an enthusiastic view of a future in which technology enables “total recall” of our lives through “total capture” of personally relevant information. Such information includes the paper and digital documents we work on or look at; email, paper mail, and instant messages sent and received; content of telephone conversations; Web sites visited; and charge-card transactions. Also included are data from other everyday activities (such as still images, video, ambient sound and location data). Finally, these personal archives might also be supplemented with environmental measures (such as light intensity and temperature variation) and even internal, biosensor data (such as heart rate and galvanic skin-response measures) reflecting our physical and emotional state.

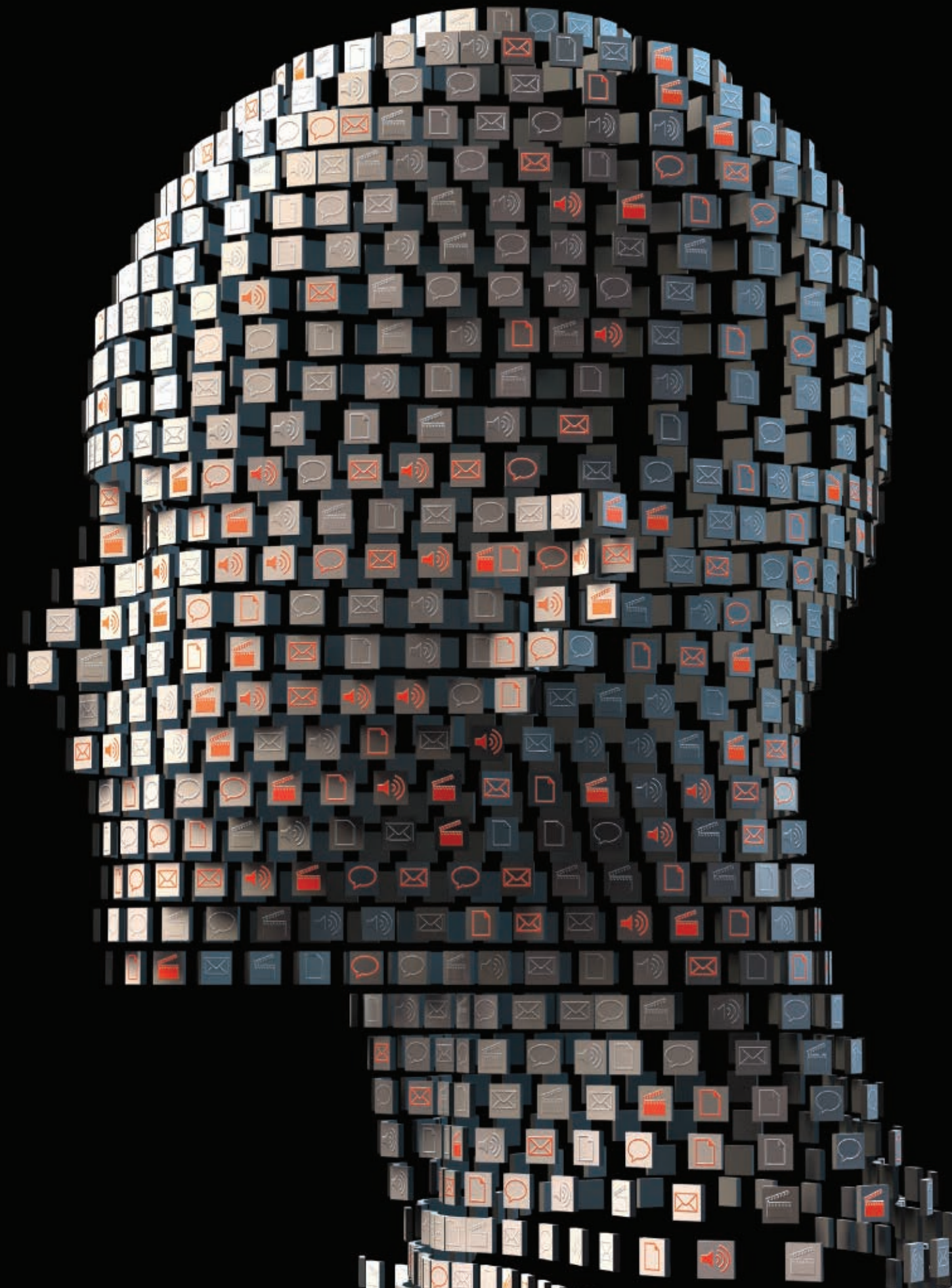
Constructing such a diverse archive of personal information requires a range of technologies for its capture, management, and storage. Today’s advances in wearable sensors, networking capabilities, and massive increases in digital-storage capacity mean this vision is feasible, fueling enthusiasm for the possibilities offered by the technology itself.

Further impetus comes from speculation about what a comprehensive lifelog might do and how it might change our lives. As outlined in 2006 by Czerwinski et al.,⁹ lifelog-

» key insights

- Focusing on “total capture,” current approaches to lifelogging have failed to explore what practical purpose such exhaustive personal digital records might actually serve.
- Evaluating new approaches, psychology has emerged as an underexploited resource in defining the nature of human memory and its key processes and weaknesses.
- Psychology as design framework could help define the types of memory such systems should support, along with their key interface properties and need to work in synergy with human memory, rather than as its replacement.

ILLUSTRATION BY GIACOMO MARCHESI



ging could permanently change how we use and share personal data, enabling us to look back over our lives or search through and organize past experiences. It could help us find lost objects, recall names, retrieve details in documents, and review discussions in meetings. It might also offer new ways of sharing data with those we care about or offer up data to interested parties. While there are knotty privacy and security implications (a huge topic, not addressed here), the potential benefits warrant substantial programs of research and development. The vision is indeed compelling. Some proponents have even said that new technologies could give each of us a comprehensive set of “digital memories” to augment or even replace their biological counterparts.²

But how sound are these goals? What benefits would the effort bring us? In reflecting on these questions we see we need a more focused and human-centered research agenda to realize the grand ambition. This, in turn, entails moving away from an obsession with “capturing everything” to a more precise specification of what it means to support human memory, leading to specific systems and concrete design implications.

History

Despite the recent upsurge of interest, lifelogging systems are not new. They can be traced back to Vannevar Bush’s 1945 “Memex” vision (a sort of desk) supporting the archiving, searching, and indexing of personal information stores revolving around documents with which we interact.⁷ Since then, it has inspired many different systems, though mainly in research laboratories. Early forays were close to the original vision, confined to capturing digital objects within an integrated system. Both past¹¹ and present¹² examples are in fact infrastructures for storing collections of heterogeneous digital objects that users generate or encounter, including documents, photos, Web pages, and email. And as infrastructures have developed, so too have the tools for searching, browsing, and retrieving information from the collections. The tools often make use of metadata about the various aspects of an object’s past context, or provenance (such as when and where it

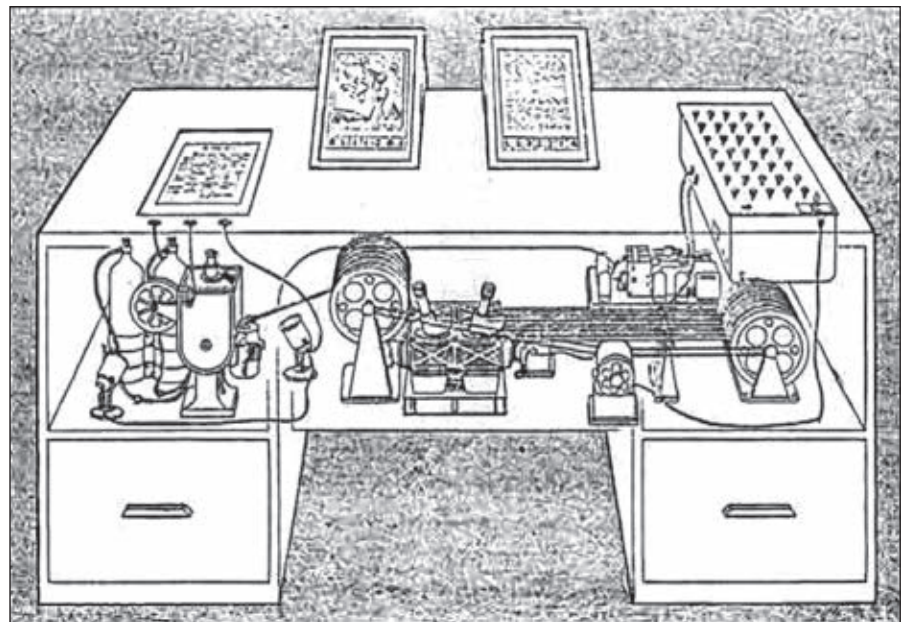
was created, who it came from, and its relationships with other objects), making it easier to re-find information (such as Dumais et al.¹⁰).

However, today’s lifelogging vision extends beyond the mere storage of desktop objects. Just as we can capture and collect personal interactions with documents, we can capture activities away from the computer, out of the office, in the everyday world. Key to it all is that many everyday activities can be captured automatically and comprehensively through digital tools that allow us to not only store important content, but also contextual details of the activities to help access the content

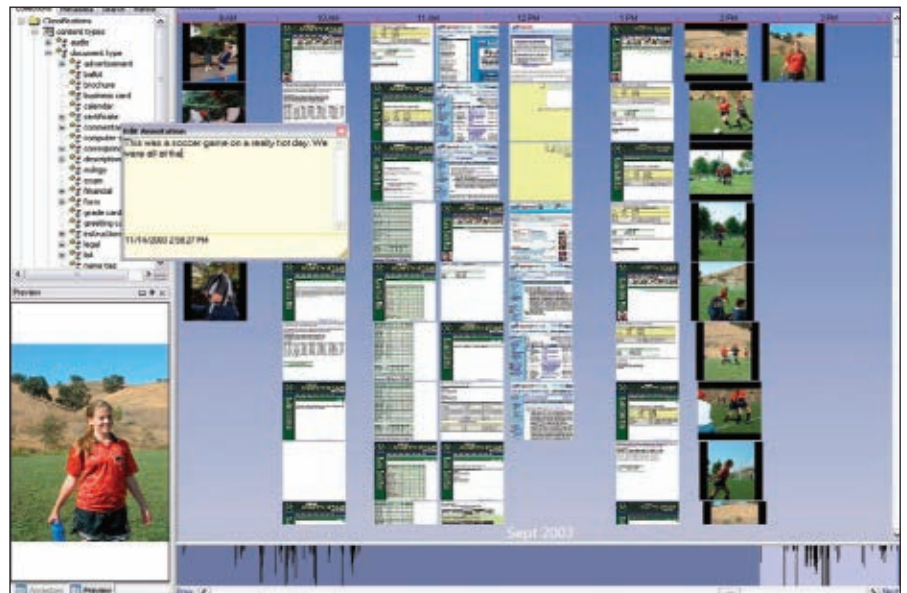
later. Note that we distinguish between lifelogging and other more deliberate activities involving the capture of personal data (such as digital photography and blogging) that involve the *effortful* selective capture and display of digital materials for a particular audience. In contrast, lifelogging seeks to be effortless and all-encompassing in terms of data capture.

There are two main classes of lifelogging system:

Total capture. For this one, the target is a complete record of everyday life, capturing as many kinds of data as possible, as continuously as possible. Data types range from documents to



Vannevar Bush's 1945 “Memex” vision.



MyLifeBits by Gordon Bell is a lifetime store of everything, aiming to fulfill the Memex vision.

images to videos to sound to location to ambient temperature, along with light levels and biosensor readings. In early systems (such as Lamming et al.¹⁷), capturing people's locations was a focus, involving users carrying or wearing small devices tracked by localized networked sensors. More recent wearable systems use head-mounted still and video cameras,¹⁵ sensor-triggered still cameras worn around the neck¹⁴ (see the figure here), and audio-capture devices.²⁶ Yet others rely on instrumented environments that capture human activity through sensors or networks, as in MIT's "PlaceLab" (http://architecture.mit.edu/house_n/placelab.html).

Situation-specific capture. These are more limited in scope, aiming to capture rich data in specific domains involving complex information. They can be viewed as a specialized form of lifelogging, aiming to record multiple kinds of data as completely and automatically as possible for specific activities or in particular places where the activity occurs. Most focus on recording activities during meetings, lectures, or other forms of work-related conversation, allowing "organizational knowledge" to be browsed and searched.²⁸ Early systems involved the simple indexing of recorded audio and pen-stroke data. More recent technology-enhanced meeting rooms capture video from multiple cameras and microphones, combining it with whiteboard content, slide capture, and digital pen strokes. Often included is software that automatically summarizes and extracts key events from the data.

Defining the Benefits: Five Rs

Surprisingly, many lifelogging systems lack an explicit description of potential value for users, focusing instead on technical challenges (such as data capture and retrieval mechanisms). When claims are made about potential benefits, they tend to be ill-defined. Nevertheless, it is important to clarify what these claims might be. Here, we outline potential benefits for memory by describing the ways such systems might support "the five Rs," or the activities we call recollecting, reminiscing, retrieving, reflecting, and remembering intentions:

Recollecting. Technology could help



Surprisingly, many lifelogging systems lack an explicit description of potential value for users, focusing instead on technical challenges.



us mentally "re-live" specific life experiences, thinking back in detail to past personal experiences (often called "episodic" memories²⁵). Remembering aspects of a past experience can serve many practical purposes; examples include locating lost physical objects by mentally retracing our steps, recollecting faces and names by recalling when and where we met someone, or remembering the details of what was discussed in a meeting.

Reminiscing. As a special case of recollection, lifelogs could also help users re-live past experiences for emotional or sentimental reasons. This can be done by individuals or socially in groups, as an (often pleasurable) end in itself; examples are watching home videos and flipping through photo albums with friends and family.

Retrieving. Lifelogs promise to help us retrieve specific digital information we've encountered over the years (such as documents, email, and Web pages). Retrieval can include elements of recollection; for example, retrieving a document might require remembering the details of when we wrote it, when we last used it, or where we put it. Alternatively, retrieval might depend on inferential reasoning (such as trying to deduce keywords in a document or thinking about the document's other likely properties, like type and size). Pondering information properties need not involve recollection of past experiences at all, as long as other ways are available for finding the desired information.

Reflecting. Lifelogs might support a more abstract representation of personal data to facilitate reflection on, and reviewing of, past experience. Reflection might include examining patterns of past experiences (such as about one's behavior over time). Such patterns might provide useful information about our general level of physical activity or emotional states in different situations, allowing us to relate it to other data about, say, our health. Alternatively, reflection might involve looking at one's past experiences from different angles or perspectives. Here, the value is not in re-living past events (as in recollecting) but in seeing things anew and framing the past differently.¹³ The value is less about memory per se than it is about

learning and self-identity.

Remembering intentions. Another type of activity concerns remembering prospective events in one's life ("prospective memory"), as opposed to the things that have happened in the past. Our everyday activities require that we constantly defer actions and plan future activities; examples include remembering to run errands, take medication, and show up for appointments.

A careful analysis of the lifelogging literature for proposed user value suggests a general focus on the processes of recollection and retrieval, but these benefits are usually implied rather than explicit.

Evaluation

Do lifelogging systems deliver these benefits? One way to identify evidence of utility is through systematic evaluation (such as focused lab studies and studies of longer-term real-world use). However, such evaluations have yet to provide overwhelming evidence of effectiveness, and extended usage studies show little impact outside research labs. Worse, few lifelogging systems are in widespread use, even within the research laboratories that developed them.

In practice, total-capture systems have been used by only a small number of people (often those with direct investment in the technology), and lifelogging infrastructures^{11,12} are not in widespread use. While Gordon Bell of Microsoft³ and Steve Mann of the University of Toronto¹⁸ have both "lived the vision" by recording many aspects of their everyday lives, they are unusual in the extreme extent of their engagement with lifelogging. Otherwise, there are few instances of full-fledged use of total capture.

More controlled system evaluations are also not encouraging in terms of demonstrable utility. The SenseCam (see the figure), which captures a series of still images based on movement and changes in the intensity of light and heat, has been shown to support the recollection of everyday experience, as well as retrieval of information about past events.²³ However, the same study showed that the capacity for these images to help people recollect their past experience rapidly decreased after only three months, casting doubt on whether such devices can support longer-term recollection.

Most evaluations described in the literature have examined situation-

specific capture systems. For example, FiloChat is an application that allows users to access spoken information from meetings via personal digital notes. Both lab experiments and a field trial at Hewlett-Packard Labs in the early 1990s demonstrated FiloChat's superiority in supporting retrieval of meeting information compared with traditional techniques (such as pen-and-paper notes and Dictaphone recordings). An active user group reported positive reactions to the system because it allowed them to generate accurate meeting minutes. Similar findings were reported in other field studies of related meeting-capture technologies; see Whittaker et al.²⁸ for a review. But despite such early positive results, more recent research should make us skeptical, suggesting that records may be less useful than we might first think. For example, lecture recordings don't significantly improve student grades,¹ and evaluations of meeting-capture systems have shown little uptake of sophisticated records.²⁰

Other research confirms the view that digital archives may be generally less valuable than people would hope. Even when—contrary to lifelogging principles—we deliberately choose to save digital memorabilia, we seldom access them. Studies by Petrelli and Whittaker¹⁹ in 2010 of digital family memorabilia (such as photos, videos, scanned images, and email) showed that digital archives are rarely accessed. In the same study, when people with large collections of digital memorabilia were asked to select objects of mnemonic significance in their homes, less than 2% of the objects they reported were digital.¹⁹ Other work has found that users with collections of thousands of digital photos never access the majority of them.²⁷

While this lack of interest in digital data doesn't imply that all such archives have little value, it raises questions about their utility in remembering or reviewing the past. One might surmise that we simply lack effective techniques for accessing the archives. However, most situation-specific capture systems for meetings and lectures include sophisticated access tools (such as multimedia search and bespoke browsing interfaces), suggesting



Microsoft SenseCam captures a series of still images triggered by changes in activity (such as movement) and includes sensors that capture other kinds of data (such as ambient light levels and temperature); courtesy Microsoft Research Cambridge.

that having poor access is likely not a general explanation. Other researchers have argued that new access tools (such as desktop search) will facilitate exploitation of digital archives.²¹ However, research at the University of Sheffield in 2008 on desktop search suggests it is used only infrequently. More important, there is no consistent evidence that improving the quality of search leads to increased use of search tools.³ These results indicate that desktop search is not a “silver bullet” leading to effective access to and exploitation of our personal digital archives.

In general, these findings imply that archival data may be less valuable than the considerable effort expended on these systems would justify.

Design Principles

How can we prevent creating under-used infrastructures or only proof-of-concept demonstrators? Needed is a new conceptual framework that is better focused on the functions lifelogging technologies could serve. Despite the memory terminology used in lifelogging work, little attention seems to focus on human memory and how it operates. Psychological studies of memory are largely ignored, even though they provide relevant concepts and results that lead directly to new design principles:

Strategically targeting the weaknesses of human memory. Total-capture systems are indiscriminate, assuming that all kinds of data are equally valuable and the more data the better. The argument often goes that we should capture “as much as we can” because we never know what we might need to remember in the future. But this “just-in-case” principle has two weaknesses: First, we can never capture absolutely everything, so choices must indeed be made when designing and building systems; for example, different kinds of data require different kinds of sensors or capture devices, adding complexity for the people using and building the systems. Second, capturing vast arrays of data might overwhelm end users maintaining and retrieving valuable information from large archives; it also ignores the burden huge amounts of data impose on system designers and developers.

Previous research provides future guidance; for example, psychology re-



Collections of digital data can serve as cues to trigger autobiographical memory about past events but are not memories in themselves.



search provides a deeper understanding of the most frequent and critical kinds of memory problems people have, allowing system designers to focus on areas of true value to users. This means that, rather than the overambitious goal of “logging everything,” creators of lifelogging systems would be better off directing their efforts at the kinds of data people find most valuable and the issues they find most problematic. In addition to the problem of the transience of memory (implicitly addressed by much lifelogging technology), people are subject to myriad other distortions and inaccuracies in memory (such as absentmindedness, blocking, misattribution, suggestibility, bias, and persistence).²² And while almost all lifelogging applications focus on supporting people’s past (retrospective) memory, strong evidence indicates that people have greater difficulty remembering what they intend to do in the future (prospective memory).²⁴ Other memory studies have identified specific groups (such as Alzheimer’s patients and aging populations) with debilitating memory conditions and demonstrated how visual-recording technologies (such as SenseCam) can be of help.⁴

Not “capturing experience” but designing effective retrieval cues. The language used by lifelog proponents sometimes conflates cueing with experiential capture. This distinction may seem obvious but is worth restating. Collections of digital data (such as sets of digital photos and sounds) can serve as cues to trigger autobiographical memory about past events but are not memories in themselves or in any way facsimiles of personal experience. Following this principle, we are thus better able to address the precise mechanisms by which cues help memory. For example, metadata can help cue retrieval of lost files by, say, providing contextual information about who wrote a document and when it was written. Alternatively, information in the digital archive may itself serve to cue a forgotten memory (such as when a stored digital photo prompts reminiscence about a previously forgotten incident).

There is again highly relevant psychology research detailing how different cues (such as time, place, people, and events) trigger autobiographical memories, suggesting (in contrast to

the design of many lifelogging user interfaces) that place, events, and people are stronger cues than time.⁵ Other research has found that cues can trigger inferences about what *must* have happened in someone's life, rather than genuine recollection²³; for example, a photo showing us having lunch with friends may cause us to truly remember the event or simply lead us to conclude we must have been there. Finally, while recollecting the past is highly dependent on the kind of cues presented to people, for prospective memory (memory for future events and intentions), the important issue is not so much the type of cue but rather when a cue is delivered, allowing an intention to be remembered at the right time and place.²⁴ This observation suggests that the capture of data (such as location or other contextual cues) might be used to trigger reminders rather than provide content to be remembered.

These observations run counter to much of the rhetoric surrounding lifelogging, where such phrases as “digital memories” and “experience capture” are often used. They show instead the importance of understanding the precise relationship between the cues we are able to capture through lifelogging technologies and the memory experiences they trigger, with clear implications for how we might design improved systems.

Support for the Five Rs. Most lifelogging proponents presume that the systems deliver benefits without being specific about what the benefits might be, assuming a one-system-suits-all approach. As we outlined earlier in the section on lifelogging benefits, the psychological literature distinguishes many different types of memory, each involving different retrieval processes. Thus, it is not simply a question of what the system captures but determining *how* such a system would be used. This determination depends largely on the type of memory being targeted or, more generally, the kind of benefit system designers hope to deliver to end users.

Many total-capture systems implicitly seem to address *recollection*, or remembering past personal experiences. It is well known in the psychological literature that there are strong connections between these autobiographical memories and visual images.⁸ This

Systems supporting reminiscence and reflection have received far less attention than those supporting recollection and retrieval.

suggests that the interfaces for such systems should focus on images as the backbone of their design. In contrast, systems for *retrieval* need not be concerned with recollection, but rather with efficient ways of searching through large heterogeneous collections of data and so provide access to metadata that might support effective search. If system designers decide to support *reminiscence*, other kinds of factors become important (such as optimizing the sharing of data with others). Systems for *reflection* might be different still where abstraction is important, offering flexible and novel methods for viewing personal data in ways that might surprise, provoke, or educate users. Finally, designing systems to support *remembering intentions* need to focus on delivering timely cues in appropriate contexts if they are to provide effective reminders.

Applying such memory taxonomies is vital for designing effective systems. First, they clarify the aspects of memory the systems are trying to support; without such clarification it is difficult to know whether the systems succeed. Second, understanding the relevant psychological processes allows designers to create systems to better support the processes. Third, taxonomies can suggest new directions for exploration; for example, systems supporting reminiscence and reflection have received far less attention than those supporting recollection and retrieval.

Offloading and metamemory. Much of the lifelogging approach is motivated by the observation that human memory is fallible. Lifelog proponents argue we need to remove the memory burden from humans, offloading it to reliable and comprehensive external digital stores. These claims need careful scrutiny, as we calculate the costs associated with digital archives, even if they make it possible to store vast amounts of data. For example, the effort required to capture, create, and maintain some kinds of data can be prohibitive. Moreover, accessing data can be inefficient compared with exploiting “organic” human memory.

How, when, and why people exploit external memory tools has been studied extensively in the psychological subfield of metamemory,⁶ which addresses people's understanding of the strengths and weaknesses of their

own memories and the strategies they employ to overcome the weaknesses. Kalnikaite and Whittaker¹⁶ looked at the interplay between organic memory and metamemory in determining when people choose to access digital conversational records, showing that even when a digital record is accurate and complete, users do not rely on it if they feel they can remember the information unaided. The decision to use a digital-memory aid also depends on the efficiency with which a memory aid can be accessed. Indeed, the study found that efficiency of access sometimes overrides accuracy, with subjects being willing to settle for less-than-perfect accuracy, as long as the method is quick and easy to use.

Lifelogging applications must be better at analyzing these trade-offs. When should people use efficient but fallible organic memory instead of less efficient but potentially more accurate digital records? Rather than seeing lifelogs as replacing memory, system designers would be better off viewing them as working in synergy with organic memory.

Lessons and Research Questions

For lifelogging research, prior work offers four key insights:

Selectivity, not total capture. Rather than unfocused efforts to “capture everything,” system designers should channel their efforts more fruitfully by identifying the situations where human memory is poor or targeting the things users most want to remember. These situations are where the systems would provide their greatest utility. Furthermore, people may sometimes want to *forget*, a view anathema to the lifelogging philosophy.

Cues not capture. System designers must be explicit about the fact that these systems do not “capture experiences” but instead provide cues that might trigger different kinds of memories. This is not a simple matter of infelicity in language, pointing instead to the need to better understand cueing processes to build systems that genuinely support user requirements for memory support.

Memory refers to a complex, multifaceted set of concepts. There are different types of memory, and system designers must clarify the aspects of memory they are targeting (such as recollection,

reminiscence, retrieval, reflection, and remembering intentions). Greater clarity should produce systems that better fit their intended purpose and support the user’s relevant cognitive processes. Unless system designers know precisely what their systems are intended to do, they can’t determine whether their designs are successful.

Synergy not substitution. Much of the rhetoric on behalf of lifelogging assumes the systems will replace human memories. However, digital records are used only when they are efficient to access and when users feel their own memory is unreliable. Rather than try to replace human memory with digital systems, system designers should look to capitalize on the strengths of human memory and help overcome its weaknesses. Lifelogging design must therefore be engineered to work in synergy with users’ own memories.

Incorporating the psychology of memory into the design of novel mnemonic technologies opens up exciting possibilities for ways to augment and support human endeavors. In light of the diversity of human memory and how it plays out in everyday life, we’ve sought to outline a design space that exploits the strong body of psychological knowledge we already have in hand. ■

References

1. Abowd, G. Classroom 2000: An experiment with the instrumentation of a living educational environment. *IBM Systems Journal (Special Issue on Pervasive Computing)* 38, 4 (Oct. 1999), 508–530.
2. Bell, G. and Gemmell, J. *Total Recall: How the E-Memory Revolution Will Change Everything*. Dutton, New York, 2009.
3. Bergman, O., Beyth-Marom, R., Nachmias R., Gradovitch, N., and Whittaker S. Improved search engines and navigation preference in personal information management. *ACM Transactions on Office Information Systems* 26, 4 (Sept. 2008), 1–24.
4. Berry, E., Kapur, N., Williams, L., Hodges, S., Watson, P., Smyth, G., Srinivasan, J., Smith, R., Wilson, B., and Wood, K. The use of a wearable camera: SenseCam as a pictorial diary to improve autobiographical memory in a patient with limbic encephalitis. *Neuropsychological Rehabilitation* 17, 4/5 (2007), 582–681.
5. Brewer, W. Memory for randomly sampled autobiographical events. In *Remembering Reconsidered: Ecological and Traditional Approaches to the Study of Memory*, U. Neisser and E. Winograd, Eds. Cambridge University Press, Cambridge, U.K., 1988, 21–90.
6. Brown, A. Metacognition, executive control, self-regulation, and other mysterious mechanisms. In *Metacognition, Motivation, and Understanding*, F.E. Weinert and R.H. Kluwe, Eds. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987, 65–116.
7. Bush, V. As we may think. *Atlantic Monthly* 176, 1 (July 1945), 101–108.
8. Conway, M. *Autobiographical Memory*. Open University Press, Milton Keynes, U.K., 1990.
9. Czerwinski, M., Gage, D., Gemmell, J., Marshall, C., Perez-Quinones, M., Skeels, M., and Catarci, T. Digital memories in an era of ubiquitous computing and abundant storage. *Commun. ACM* 49, 1 (Jan. 2006), 44–50.
10. Dumais, S., Cutrell, E., Cadiz, J., Jancke, G., Sarin R., and Robbins D. Stuff I’ve seen: A system for personal information retrieval and re-use. In *Proceedings of the Conference of the Special Interest Group on Information Retrieval* (Toronto, 2003). ACM Press, New York, 2003, 72–79.
11. Freeman, E. and Fertig, S. *Lifestreams: Organizing your electronic life*. In *Proceedings of the AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval* (Cambridge, MA, Nov.). AAAI Press, Menlo Park, CA, 1995.
12. Gemmell, J., Bell, G., and Lueder, R. MyLifeBits: A personal database for everything. *Commun. ACM* 49, 1 (Jan. 2006), 88–95.
13. Harper, R., Randall, D., Smyth, N., Evans, C., Heledd, L., and Moore, R. The past is a different place: They do things differently there. In *Proceedings of the Conference on Designing Interactive Systems*. ACM Press, New York, 2008, 271–280.
14. Hodges, S., Williams, L., Berry, E., Izadi, S., Srinivasan, J., Butler, A., Smyth, G., Kapur, N., and Wood, K. SenseCam: A retrospective memory aid. In *Proceedings of Ubicomp* (Orange County, CA, Sept.). Springer, Berlin, 2006, 177–193.
15. Hori, T. and Aizawa, K. Context-based video retrieval system for the lifelog applications. In *Proceedings of the ACM Workshop on Multimedia Information Retrieval* (Berkeley, CA, Nov.). ACM Press, New York, 2003, 31–38.
16. Kalnikaite, V. and Whittaker, S. Software or wetware? Discovering when and why people use digital prosthetic memory. In *Proceedings of the Conference on Human Factors in Computing Systems* (San Jose, CA, Apr.–May). ACM Press, New York, 2007, 71–80.
17. Lamming, M., Brown, P., Carter, K., Eldridge, M., Flynn, M., Louie, G., Robinson, P., and Sellen, A. The design of a human memory prosthesis. *Computer Journal* 37, 3 (Jan. 1994), 153–163.
18. Mann, S. Wearable computing: A first step toward personal imaging. *Computer* 30, 2 (Feb. 1997), 25–32.
19. Petrelli, D. and Whittaker, S. Family memories in the home: Contrasting physical and digital mementos. *Personal and Ubiquitous Computing* 14, 2, (Feb. 2010), 153–169.
20. Richter, H., Miller, C., Abowd, G., and Funk, H. Tagging knowledge acquisition to facilitate knowledge traceability. *International Journal on Software Engineering and Knowledge Engineering* 14, 1 (Feb. 2004), 3–19.
21. Russell, D.M. and Lawrence, S. Search everything. In *Personal Information Management*, W. Jones and J. Teevan, Eds. University of Washington Press, Seattle, 2007.
22. Schacter, D. *The Seven Sins of Memory: How the Mind Forgets and Remembers*. Houghton Mifflin, New York, 2001.
23. Sellen, A., Fogg, A., Hodges, S., Rother, C., and Wood, K. Do lifelogging technologies support memory for the past? An experimental study using SenseCam. In *Proceedings of the Conference on Human Factors in Computing Systems* (San Jose, CA, Apr.–May). ACM Press, New York, 2007, 81–90.
24. Sellen, A., Louie, G., Harris, J., and Wilkins, A. What brings intentions to mind? An in situ study of prospective memory. *Memory* 5, 4 (July 1997), 483–507.
25. Tulving, E. *Elements of Episodic Memory*. Oxford University Press, New York, 1983.
26. Vermuri, S., Schmandt, C., Bender, W., Tellex, S., and Lassey, B. An audio-based personal memory aid. In *Proceedings of Ubicomp* (Nottingham, U.K., Sept.). Springer, Berlin, 2004, 400–417.
27. Whittaker, S., Bergman, O., and Clough, P. Easy on that trigger dad: A study of long-term family photo retrieval. *Personal and Ubiquitous Computing* 14, 1 (Jan. 2010), 31–43.
28. Whittaker, S., Tucker, S., Swamplai, K., and Laban, R. Design and evaluation of systems to support interaction capture and retrieval. *Personal and Ubiquitous Computing* 12, 3 (Mar. 2008), 197–221.

Abigail J. Sellen (asellen@microsoft.com) is a principal researcher at Microsoft Research in Cambridge, U.K., and Special Professor of Interaction at the University of Nottingham, U.K.

Steve Whittaker (whittaker@almaden.ibm.com) is a research scientist at IBM Almaden Research Center, San Jose, CA.

© 2010 ACM 0001-0782/10/0500 \$10.00

DOI:10.1145/1735223.1735244

The curriculum should inspire students to view CS as both accomplishment and intellectual discipline.

**BY CLAYTON LEWIS, MICHELE H. JACKSON,
AND WILLIAM M. WAITE**

Student and Faculty Attitudes and Beliefs About Computer Science

WHAT STUDENTS THINK about a discipline—its structure, usefulness, how it is learned—plays an important role in shaping how they approach it. Just as faculty members aim to have their students learn the facts and skills of a discipline, they may also want to shape student beliefs and attitudes. Here, we report the attitudes of undergraduate computer science students early and late in the curriculum, comparing them with faculty attitudes in the same department. The results reflect the places where students think

what faculty want them to think, where they do not think that way, and whether there is evidence that final-year students agree more or less with faculty than students in introductory courses. Together with earlier research, the results provide insight into sometimes surprising attitudes, and can help guide curricular improvement.

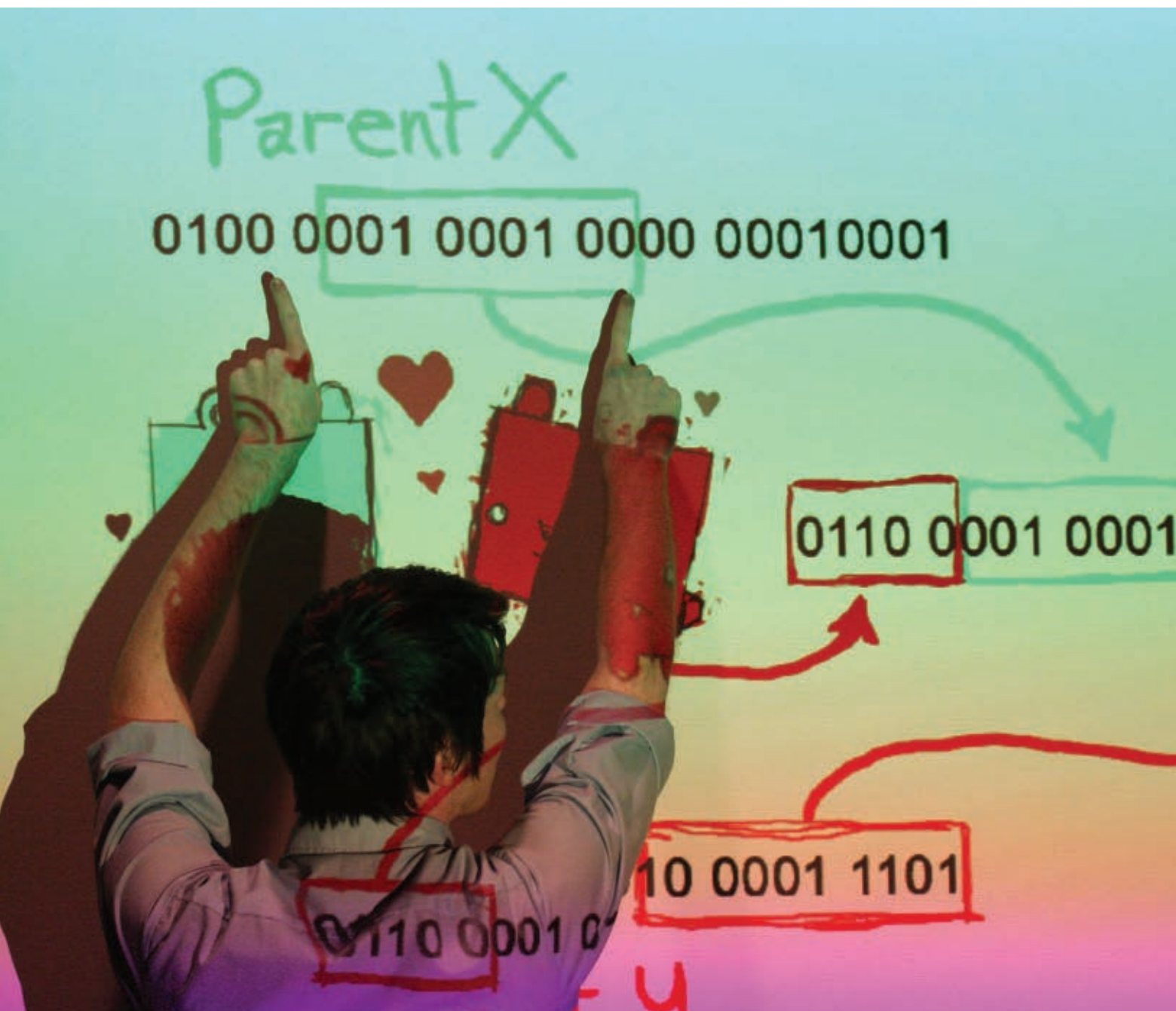
In physics education, research^{1,13} into key attitudes and beliefs about physics as a discipline and how they change suggests that courses sometimes shift student attitudes away from the attitudes endorsed by faculty. In particular, students may move toward the view that physics is mainly about memorizing and evaluating formulae, rather than about a conceptual understanding of the natural world.

CS faculty are also likewise concerned with student attitudes: “CS is just programming;”¹⁷ “As long as a program works it doesn’t matter how it is written;” and “Theoretical CS is not relevant to the real world.” Do students hold such views? As they move through the curriculum, do their beliefs come to resemble those of faculty teaching them?

Our study collected and analyzed data on these points. Specifically, we collected responses to 32 questions about attitudes and beliefs from beginning and advanced CS undergraduates and from faculty at the University of Colorado at Boulder. The results revealed some areas in which student responses clearly agree with faculty and others where they disagree. Comparing the responses of beginning

» key insights

- **The attitudes of beginning CS students are more varied than final-year students, suggesting the curriculum plays an important role in shaping them.**
- **Final-year CS students generally show little appreciation for skills involving creativity and reasoning, emphasizing instead CS as outcomes.**
- **Understanding how student attitudes are formed and strengthened helps faculty develop more effective CS curricula.**



students with those of more advanced students also suggests how progress through the curriculum changes student beliefs and whether such change is toward or away from what faculty believe.

We gathered 38 survey items from four sources: the Colorado Learning Analysis Survey for physics,¹ selected on the grounds they were relevant (or relevant with small changes) to CS; an interview study of student attitudes toward group work in the CS curriculum¹⁶; faculty suggestions on student attitudes causing them concern; and

Carol Dweck's work⁷ on student attitudes about the roles of aptitude and effort in academic success.

We discussed the 38 items with a group of six volunteer students to determine whether they were easily understood. We discarded six items based on student input and changed the wording of others to make the intent of the survey statements clearer. The final survey consisted of 32 items; the table here includes the item numbers (non-consecutive) we used in coding the data and in analyzing the results.

Each of the 32 items asked faculty

to indicate whether they strongly disagreed, disagreed, neither agreed nor disagreed, agreed, or strongly agreed. We gave students the same options but also asked them what they thought a CS professor would want them to say; for this response they were also allowed to indicate "don't know." The survey included a "catch" item (for both faculty and students) that should be left intentionally blank so responses from participants who simply filled in responses without reading the items could be discarded.

In the final week of the spring 2006

Survey results

Item #		Faculty	% Agree with Faculty Response			Senior Agreement with Faculty	Change Seniors vs CS1
			CS1	CS2	Senior		
Cluster 1: Computer Science as Accomplishment							
Subcluster: Don't just learn from the examples.							
10	There is usually only one correct approach to solving a computer science problem.	Reject	91	98	95	☺	4
40	To learn computer science, I only need to memorize solutions to sample problems.	Reject	80	94	90	☺	10
54	Reading books or manuals is usually a waste of time in computer science.	Reject	75	73	61	☹	-14
Subcluster: The end justifies the means.							
20	On a computer science assignment, what matters is getting the desired result, not how you arrive at the result.	Reject	42	51	46	☹	4
52	If a program works it doesn't matter much how it is written.	Reject	55	73*	80	☺	25*
68	Doing things the "right" way isn't as important as just pushing through to a solution.	Reject	44	55	68	☹	24*
Theme: Work in the real world.							
16	In the real world, computer scientists spend a lot of time working alone.	Reject	51	65	66	☹	15
18	In the real world, computer scientists' work is mostly programming.	Reject	25	57*	76	☺	51*
62	Managers are usually people who can't cut it technically.	Reject	46	47	58	☹	12
Theme: Group work.							
8	If you work in a group you can't take as much pride in the results as when you work alone.	Reject	62	53	73	☹	11
36	You can only truly understand something if you figure it out on your own.	Reject	51	53	51	☹	0
Other							
6	Getting through the computer science curriculum basically means being able to recall something you've read or been shown.	Reject	48	55	54	☹	6
14	I cannot learn computer science without a teacher to explain it.	Reject	51	57	63	☹	12
28	A significant problem in learning computer science is being able to memorize all the information I need to know	Reject	24	51	68	☹	44*
56	In the computer science curriculum, if you spend enough time coding you'll get all you need to get from the courses.	Reject	27	45	73	☹	46*
66	If you know what you are doing you can leave work to the last minute and still get it done.	Reject	46	53	49	☹	3
Cluster 2: Computer Science as an Intellectual Discipline							
Subcluster: Computer Science is creative and valuable.							
50	The work you do in computer science in the real world requires a lot of creativity.	Endorse	83	82	68	☹	-15
58	Research in computer science often develops really important ideas.	Endorse	56	84*	66	☹	10
60	Reasoning skills used to understand computer science material can be helpful to me in understanding things in everyday life.	Endorse	70	80	61	☹	-9
Related items: Concepts and understanding are important.							
12	I am not satisfied until I understand why something works the way it does.	Endorse	79	82	80	☺	1
22	Theoretical computer science, such as analysis of algorithms, isn't very relevant to the real world.	Reject	73	75	56	☹	-17
44	When I solve a computer science problem, I explicitly think about which computer science ideas apply to the problem.	Endorse	72	65	44	☹	-28*
46	If I get stuck on a computer science problem, there is no chance I'll figure it out on my own.	Reject	59	84*	85	☺	26*
64	If you can do something you don't need to understand it.	Reject	79	80	80	☺	1
Other							
4	Nearly everyone is capable of succeeding in the computer science curriculum if they work at it.	Reject	30	26	34	☹	4
26	After I study a topic in computer science and feel that I understand it, I have difficulty solving problems on the same topic.	Reject	45	57	61	☹	16
48	There are times I solve a computer science problem more than one way to help my understanding.	Endorse	41	49	61	☹	20*
Items on which faculty disagreed							
24	The curriculum in computer science covers many disconnected topics.						
30	I find that reading the textbook carefully is a good way for me to learn the topics covered in my computer science courses.						
32	It's better to jump into a problem instead of trying to analyze it first.						
34	It's a lot more satisfying to do an assignment on your own than with help.						
38	When you submit a group project, the instructor can't tell how good your individual work was.						

semester, we administered the survey by email to faculty and in paper form to students in three CS courses: first-semester introductory (CS1), second-semester introductory (CS2), and senior-level capstone design.

Discussion

We obtained responses from 13 faculty (of a total of 25). For student surveys, we received 71 surveys from CS1, 48 from CS2, and 41 from senior capstone. The survey was voluntary, though no more than one or two students in each class declined to participate. No surveys contained a response to the catch item, but we did reject one survey because the last three pages had identical responses for each item.

We tallied responses by grouping “strongly disagree” and “disagree” as negative responses, “strongly agree” and “agree” as positive responses, and all other responses, including omitted responses, as neutral. We examined the responses by faculty to classify the items as either rejected or endorsed by faculty. Using the criterion that 75% or more of faculty had to agree to reject or endorse an item, we excluded five items as not showing consensus among the faculty (see cluster 2 in the table).

We placed the remaining 27 items in thematic categories using a combination of what Adams et al.¹ called “predeterminism” and “raw statistical” grouping. We first sorted them into groups reflecting our sense of the relationships among them, without reference to the data (predeterminism) and used hierarchical cluster analysis, a statistical technique, to identify items participants commonly responded to in the same way (using the SPSS 16 package²).

Before we performed cluster analysis, we transformed responses for items in the same thematic category so answers reflecting a related underlying attitude would be coded the same. For example, we transformed the responses to item 64 (“If you can do something you don’t need to understand it”) so a negative response would match a positive response to item 12 (“I am not satisfied until I understand why something works the way it does”).

We used the results of the cluster analysis to modify the groupings to

bring the resulting categories in line with the data, where appropriate. That is, where the data showed the participants commonly answered two items the same way, we grouped these items together, even if they were not grouped together in our original classification. In other cases, where the data showed that two items we thought were related were actually commonly answered differently, we adjusted the grouping to reflect that fact.

The table shows the resulting groupings of items. At the highest level, they fall into two broad clusters: “CS as an accomplishment” and “CS as an intellectual discipline.” Within them, “subclusters” are groups of items that were statistically related, while “themes” were items with related content not strongly related in the data. For example, items 8 and 36 both relate to aspects of group work, so they are thematically related, but participants often gave different responses to them. In cluster 2, the items in the subcluster were closely related in the data; the “related” items formed a larger cluster around them and were less related to the “other” items in the cluster.

We marked each item in the table for which faculty shared consensus to show whether faculty rejected or endorsed the item. The percentage of students in CS1 and in CS2 and seniors (final-year students) who agreed with the faculty consensus is also shown. For most items, the percentage of CS1 and CS2 students endorsing the faculty position did not differ significantly; an asterisk next to the percentage for CS2 indicates the percentage is significantly different at the 5% level from the percentage for CS1 (according to a two-tailed Fisher’s exact test, a test for judging differences between groups when using small samples).

A face symbol indicates whether the seniors’ responses were one of the following: *generally in line with faculty responses* (75% or more of seniors endorsing the faculty position), marked by a happy face; *mixed* (between 50% and 75% of seniors endorsing the faculty position), marked by a neutral face; or *substantially in conflict* (less than 50% of seniors endorsing the faculty position), marked by a sad face.

The last column in the table (se-

niors vs. CS1) shows how senior responses compared to the responses of CS1 students. The numbers are the difference between the percentage of seniors and the percentage of CS1 students agreeing with the faculty consensus on each item. A negative number means seniors agreed with faculty less than CS1 students. Asterisks mark differences in responses significant at the 5% level (Fisher’s exact test, two-tailed).

Now consider an example of how the table reflects these aspects of the data. In the first data line (item 10), the entry shows (reading from left) that the faculty consensus rejected this item, and that 91% of CS1 students, 98% of CS2 students, and 95% of seniors also rejected the item, that the responses of the seniors were generally in line with the faculty consensus (the happy face means 75% or more agreement), and that the agreement between seniors and faculty (95%) was four percentage points greater than the agreement between CS1 students and faculty (91%).

Does greater agreement with faculty always represent goodness? No. We suggest later at least one item (4) on which we feel students were more right than faculty; readers might judge other items the same way. But faculty design the curriculum, and alignment with their attitudes is what they aim for. Failure to find alignment indicates a problem. Faculty weren’t of one mind, of course, and we removed items on which they didn’t agree, as described earlier.

In considering which items show evidence of progress or lack of progress as students move through the curriculum, we must urge caution in basing inferences on how students in the introductory classes compare with seniors; different students completed the survey in each of the classes. There were no doubt differences among the responses not traceable to having gone through more or less of the curriculum. Many surveyed CS1 students were not CS majors. A good many students join the CS program after CS1; of the senior class we surveyed, 45% joined after CS2 and 37% after CS1 and before CS2. Students also drop out of the program; about one-eighth of the surveyed CS1 students eventu-

ally ended up as CS majors. But the CS1 students reasonably represent the students available as input to the program. A finding that students late in the curriculum agree less with faculty could be due to many factors but likely shows that the output from the program is not improved with respect to the available input—a challenge to faculty charged with managing the program. The table does not present student responses about how faculty would like them to respond to each item. In some cases we bring this information into the discussion of individual items.

Among the 27 items for which we found strong faculty consensus, seniors were generally in line with faculty (75% or more agreement) on only seven items. For 16 items, we found mixed agreement (50%–75%), and for four items we found less than 50% of seniors endorsed the faculty position. Though this level of agreement was not good, it is better than for CS1 students. For 22 of the 27 items, we found seniors were more likely to agree with the faculty consensus than CS1 students. For seven of the items the difference in agreement was statistically significant. Among the five items on which seniors agreed less with the faculty consensus than CS1 students, the difference was statistically significant for one item (44).

Results by Subcluster and Theme

We now examine the data for the thematic groups of items and the subclusters to see what it says about student beliefs and how these beliefs differ between beginning and more advanced students.

Don't learn just from examples. In this statistical cluster, seniors agreed with faculty that there is more to CS than memorizing solutions or learning a single best approach to problems. They agreed less strongly that reading is important, and it may be they value reading less than, say, CS1 students.

The end justifies the means. Most seniors agreed with faculty that how a program is written is important (item 52) and were significantly more likely to share this attitude than CS1 students. This is good news. But only 68% of seniors agreed with faculty that

doing things right is more important than just getting a solution, though this response represents a significant improvement over the position of CS1 students. Less than half of the seniors felt that how they do the work on an assignment is more important than getting “the desired answer” (item 20). This is a little better, though not much, than the response from CS1 students. Most seniors (76%) correctly indicated that faculty would disagree with the item, so this was not a case of students not knowing where faculty stood on an issue.

Why do students not feel that how they complete an assignment is important? This attitude may connect with a student's wish to increase the difficulty of assignments as a way to demonstrate competence. Leonardi et al.⁸ found this tendency in interviews with CS students, writing:

“We found no place in the data where informants suggested that ignoring instructions would help them arrive at a better solution. Rather, they admitted readily that following pre-specified instructions made the task easier. But ignoring instructions increased the challenge and introduced a higher level of risk. As one engineer observed, ‘if you can do it by figuring it out yourself instead of following some cookie-cutter process then you’re on your way to becoming an expert.’ Success under such conditions demonstrated expertise and technical competence.”

Responses on this item might reflect what students saw as the role of assignments. According to some research,⁵ students may consider work on assignments to be a product offered for payment (in the form of a grade) rather than as a learning experience. Viewed this way, doing the work a certain way to learn new techniques is irrelevant.

Work in the real world. While most seniors recognized that computer scientists don't spend a lot of time working alone, a third did not. Most rejected the cartoon stereotype of the incompetent manager, though many did not. While there is room for improvement, results on these items were better than for CS1 students. Most seniors rejected item 18 (“In the real world, computer scientists' work is mostly

programming”), a matter important to many people concerned about public perceptions of the field.¹⁷ The situation was significantly better among the seniors than among CS1 students; indeed, the situation among CS2 students was also significantly better than among CS1 students, though because a good many students entered the program at CS2 this improvement cannot be attributed to what happens in the curriculum. This item also showed the greatest difference between seniors and CS1 students on the survey. Still, just less than a quarter of the seniors thought the work of computer scientists is mostly programming.

Group work. While students generally recognized that much work in computing is collaborative, and departments and programs strive to provide opportunities for students to develop collaboration skills, the results we found for the two items in the “group work” theme showed there is room for progress. More than a quarter of the seniors indicated that one can take more pride in individual work, and almost half felt that understanding must be developed alone.

Waite et al.¹⁶ and Leonardi et al.⁸ discussed negative student attitudes toward group work, reporting that many students form the misconception that individual work is the essential measure of competence for skilled professionals. Faculty did not show consensus in rejecting item 36 (“It's a lot more satisfying to do an assignment on your own than with help”) and item 38 (“When you submit a group project, the instructor can't tell how good your individual work was”), suggesting that faculty, as well as students, need to do some work in this area.

Other items in cluster 1. Items 6, 14, 28, and 56 might have been expected to appear in one of the table's subclusters (perhaps in “concepts and understanding”) but did not show response patterns closely related to the items in these clusters. As a group, they showed only middling agreement between seniors and faculty, with agreement by seniors being somewhat greater than by CS1 students. For item 28 (“A significant problem in learning computer science is being able to memorize all the information I need to know”) and item 56 (“In the computer sci-

ence curriculum, if you spend enough time coding you'll get all you need to get from the courses"), seniors agreed with faculty significantly more than CS1 students agreed with faculty.

Less than half of the seniors rejected item 66 ("If you know what you are doing you can leave work to the last minute and still get it done"), confirming a problem identified by Waite et al.¹⁶ and Leonardi et al.⁸ that students see procrastination not as a failing but as something positive, a way to demonstrate personal prowess, writing:

"The important point here is that for the informants in this study, waiting until the last minute to begin a project was not a sign of laziness or disinterest in the subject matter. Rather, beginning an assignment late makes successfully completing the task more difficult, and, thus, is a sign of their expertise and mastery of technical skill. In the laboratories on days before large projects were due, informants regularly discussed the status of their projects with one another, comparing how much they had completed. Similarly, on days on which a large project was due, student engineers typically asked one another 'When did you start?' and 'When did you finish?' Higher status was awarded to those who could wait the longest and still complete the project successfully."

Senior attitudes on this matter were hardly better than those of CS1 students and slightly worse than CS2 students; at least they weren't *much* worse.

CS is creative and valuable. Responses to the three items in this subcluster were strongly related. Unfortunately, agreement between seniors and faculty was not strong for any of them. Worse, for two items, 50 ("The work you do in computer science in the real world requires a lot of creativity") and 60 ("Reasoning skills used to understand computer science material can be helpful to me in understanding things in everyday life"), seniors agreed less with faculty than did CS1 students. For item 58 ("Research in computer science often develops really important ideas"), agreement with faculty was somewhat stronger among seniors than among CS1 students but at 66% did not constitute a ringing endorsement.



Only 68% of seniors agreed with faculty that doing things right is more important than just getting a solution, though this response represents a significant improvement over the position of CS1 students.



Related items: Concepts and understanding matter. As mentioned earlier, these items form a larger subcluster together with the subcluster just discussed. There was variation in the number of seniors endorsing the faculty consensus, pointing to the value of concepts and understanding. For two of the apparent bright spots, item 12 ("I am not satisfied until I understand why something works the way it does") and item 64 ("If you can do something you don't need to understand it"), agreement was good but hardly better among the seniors than among CS1 students. Only for item 46 ("If I get stuck on a computer science problem, there is no chance I'll figure it out on my own") was the greater agreement by seniors than by CS1 students statistically significant.

On the negative side, for two of the items in this group the seniors agreed less with faculty than did the CS1 students. For example, seniors were less likely to endorse item 44 ("When I solve a computer science problem, I explicitly think about which computer science ideas apply to the problem") than were the CS1 students. Most seniors (88%) said faculty explicitly endorsed thinking about ideas, but they themselves didn't endorse it. Why didn't they?

Interviews reported by Leonardi et al.⁸ may shed light on this misalignment, identifying a "norm" among students that "expertise is measured by task difficulty" among the students aspiring to be engineers, writing:

"The norm suggests that engineers should place value on overcoming challenge and 'beating the odds.' The work practices reflecting this norm artificially and purposefully increased the difficulty of a given task, such as a homework assignment. Taken together, these practices introduced a sense of 'sport' to engineering work by providing handicaps that ultimately decreased an informant's chances of success. Informants perceived that completing a task with a handicap was a mark of an 'expert engineer.' "

Leonardi et al. also suggested that one way students increase the difficulty of assignments (so as to demonstrate their skill to themselves and sometimes to their peers) is to ignore concepts that would actually help with the work.

Other items in cluster 2. Like some of the “other” items in cluster 1, items 26 and 58 in this group might have been expected to appear in one of the subclusters but did not. They show only middling agreement between seniors and faculty, with agreement by seniors greater than by CS1 students. For item 48 (“There are times I solve a computer science problem more than one way to help my understanding”), seniors agreed with faculty significantly more than CS1 students agreed with faculty.

Item 4 (“Nearly everyone is capable of succeeding in the computer science curriculum if they work at it”) reflects an interesting situation. Faculty consensus rejects Dweck’s view⁷ that effort is the key to success, but most seniors do not reject this attitude, only a few more than among CS1 students. For someone agreeing with Dweck, it’s good that student views on the value of effort aren’t changed much. It’s also interesting that seniors wrongly believed faculty endorse Dweck’s position, with 88% of seniors indicating that faculty would want them to agree with the item.

The data further suggests that item 4 was not very strongly related to any of the other items in the survey. Despite falling in cluster 2 in the hierarchical-clustering results, it is the item in that cluster that is least closely related to the other items.

Top-level clusters. The hierarchical cluster analysis revealed two clear categories in the data, and a review of the items in each cluster showed them to be meaningful groupings. The groups suggest that students conceptualize CS in two distinct ways: The first is “CS as accomplishment,” in which the emphasis is on outcomes and what it takes to reach them, including skill, technical expertise, programming knowledge, and resources (books, peers, teachers). The second is “CS as intellectual discipline,” in which the emphasis is on how CS offers a way to approach and understand the world, including how to reason, gain understanding and deep learning, appreciate the importance of creativity, and dwell on problems to be able to explore them fully. This intellectual-discipline view is very much the perspective on the field emphasized by Wing¹⁷ in her



Faculty must consider ways to move students toward the idea that “The work you do in computer science in the real world requires a lot of creativity,” rather than away from it.



work on computational thinking.

The fact that these two clusters emerged from the data is important. Interestingly, earlier research discussed a similar contrast between accomplishment and creativity in engineering.^{3,14,10} It is possible that the two perspectives—CS as accomplishment and CS as intellectual discipline—could be in tension with one another. How might they be reconciled or otherwise aligned?

We can revisit some of the data reviewed earlier and consider how it reflects on these perspectives. Seniors were in conflict with faculty on two items in cluster 1, and the responses from CS1 and CS2 students were similar. First, seniors believed that waiting until the last minute is acceptable if you have the know-how (item 66). Second, they believed that getting the desired result is more important than how you get there (item 20). These results directly confirm the findings of earlier research,^{8,15,16} highlighting the emphasis on accomplishment at the expense of other considerations that might be important to faculty or to effective learning.

In cluster 2 there was conflict with faculty on item 44 (“When I solve a computer science problem, I explicitly think about which computer science ideas apply to the problem”). Faculty and CS1 students agreed that they intentionally reflect on which CS ideas apply to the problem they are trying to solve. Less than half of seniors claimed to do so. This, too, supports the view of CS as competence, where skill is the application of knowledge, rather than a type of reasoning or discipline. This item (44) was the one with the greatest difference between CS1 students and seniors, in the negative direction.

The only other conflicting item is potentially troubling if we are concerned with access to the CS major. Faculty did not endorse the statement that anyone could succeed at CS if they worked at it (item 4); students in all groups consistently disagreed with faculty on this.

Looking at differences between seniors and CS1 students with respect to their agreement with faculty, with the exception of item 54, on the importance of reading, all items for which

seniors agreed less with faculty than CS1 students were in cluster 2. Compared to CS1 students, fewer seniors believed “real-world” CS requires creativity (item 50); fewer believed that either the reasoning skills of CS (item 60) or its theoretical concepts (item 22) were relevant to everyday life, and (as we discussed), fewer still were intentionally reflective when solving CS problems (item 44).

Overall, the average agreement between seniors and faculty was 67% for cluster 1 and 63% for cluster 2, not very different. But the average increase in agreement with faculty, comparing seniors with CS1 students, was 16 percentage points for cluster 1 and only one percentage point for cluster 2. The results suggest the curriculum falls significantly short in helping students develop the perspective that CS is an intellectual discipline.

Conclusion

The survey results and analysis suggest a number of challenges to the curriculum in which the students and faculty participated. Using one item (50) as an illustration, faculty must consider ways to move students toward the idea that “The work you do in computer science in the real world requires a lot of creativity,” rather than away from it. A next step could be collecting longitudinal data from the same students as they move through the curriculum. Collecting data in key courses at the beginning and end of a semester would also be useful in separating the effects of selection from the effects of courses themselves and in zeroing in on the effectiveness of courses intended to promote particular attitudes and beliefs.

Besides being important learning targets in themselves, the attitudes and beliefs explored here may also be important in other ways. Studies in university physics education show that student attitudes and beliefs relate to performance on content assessments.^{1,13} Studies in physics education also show direct evidence of selection, rather than a change in attitude, as more advanced students are compared with beginners. This selection effect raises the possibility that understanding student attitudes and beliefs could be important in terms

of retention and understanding why some groups are less well represented than others in CS programs. Although we did not collect data on gender, it is possible that attitudes and trends differ for male and female students, and that understanding them could help address underrepresentation of women in CS.

Development of attitudes and beliefs as learning goals is a key part of the process by which college education socializes students into their professions.^{4,11,12,18} Waite et al.^{15,16} presented curricular and pedagogical innovations pointing in the right direction on a number of issues (such as increasing student involvement during class and creating team-based assignments that require genuine collaboration rather than a “hands off” approach). For example, reducing the weight placed by faculty on assignment grades and encouraging collaboration can improve student attitudes toward assignments.¹⁵ Such innovation could make a big difference if adopted and reinforced throughout a curriculum. What kind of force is needed to make it happen?

“Curiosity” is one possible answer. If you are a CS faculty member, how would your students respond to the survey? Its results came from just one department, likely not yours. You might think your students would never deliberately make their work more difficult or that they are all aware of the value of CS research. But are you sure?

Curiosity is very important in fueling improvements in physics education, as faculty found their students did not respond as they would have wished on evaluation instruments shared across institutions (see, for example, Crouch and Mazur⁶). Data on this problem provided the foundation for efforts that produced measurable improvements. Can CS faculty achieve the same results?

Acknowledgements

An earlier report of these results appeared in 2007 in the *ACM SIGCSE Bulletin*.⁹ We thank Amer Diwan, Christian Doerr, Noah Finkelstein, Gary Nutt, Steven Pollock, Bruce Sanders, and Brian Shuckey for ideas, inspiration, and assistance. This project was an activity of the President’s Teaching

and Learning Collaborative (<http://www.colorado.edu/ptsp/ptlc/>) of the University of Colorado. 

References

- Adams, W.K., Perkins, K.K., Podolefsky, N., Dubson, M., Finkelstein, N.D., and Wieman, C.E. New instrument for measuring student beliefs about physics and learning physics: The Colorado Learning Attitudes about Science Survey. *Physical Review Special Topics: Physics Education Research* 2, 1 (2006).
- Anderberg, M.R. *Cluster Analysis for Applications*. Academic Press, New York, 1973.
- Bucciarelli, L.L. Designing and learning: A disjunction in contexts. *Design Studies* 24, 3 (May 2003), 295–311.
- Bucciarelli, L.L. and Kuhn, S. Engineering education and engineering practice: Improving the fit. In *Between Craft and Science: Technical Work in U.S. Settings*, S.R. Barley and J.E. Orr, Eds. ILR Press, Ithaca, NY, 1997, 210–229.
- Button, G. and Sharrock, W. Project work: The organisation of collaborative design and development in software engineering. *Computer Supported Cooperative Work* 5, 4 (Dec. 1996), 369–386.
- Crouch, C.H. and Mazur, E. Peer instruction: 10 years of experience and results. *American Journal of Physics* 69, 9 (Sept. 2001), 970–977.
- Dweck, C.J. *Self-Theories: Their Role in Motivation, Personality, and Development*. Psychology Press, Philadelphia, PA, 2000.
- Leonardi, P.M., Jackson, M.H., and Diwan, A. The enactment-externalization dialectic: Rationalization and the persistence of counterproductive practices in student engineering. *Academy of Management Journal* 52, 2 (Apr. 2009), 400–420.
- Lewis, C. Attitudes and beliefs about computer science among students and faculty. *SIGCSE Bulletin* 39, 2 (June 2007), 37–41.
- Margolis, J. and Fisher, A. Geek mythology. *Bulletin of Science, Technology, and Society* 23, 1 (Feb. 2003): 17–20.
- National Academy of Engineering. *The Engineer of 2020: Visions of Engineering in the New Century*. National Academies Press, Washington, D.C., 2004.
- Ondrack, D.A. Socialization in professional schools: A comparative study. *Administrative Science Quarterly* 20, 1 (Mar. 1975), 97–103.
- Perkins, K.K., Adams, W.K., Pollock, S.J., and Wieman, C.E. Correlating student beliefs with student learning using the Colorado Learning Attitudes About Science Survey. In *Proceedings of the Physics Education Research Conference* (Sacramento, CA, Aug. 5). AIP, Melville, NY, 2004, 61–64.
- Vincenti, W.G. *What Engineers Know and How They Know It: Analytical Studies from Aeronautical History*. Johns Hopkins University Press, Baltimore, MD, 1990.
- Waite, W.M., Jarrahan, A., Jackson, M.H., and Diwan, A. Design and implementation of a modern compiler course. In *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (Bologna, Italy, June 26–28). ACM Press, New York, 2006, 18–22.
- Waite, W.M., Jackson, M.H., Diwan, A., and Leonardi, P.M. Student culture vs. group work in computer science. *SIGCSE Bulletin* 36, 1 (Mar. 2004), 12–16.
- Wing, J. Computational thinking. *Commun. ACM* 43, 3 (Mar. 2006), 33–35.
- Yurtseven, H.O. How does the image of engineering affect student recruitment and retention? A perspective from the USA. *Global Journal of Engineering Education* 6, 1 (2002), 17–23.

Clayton Lewis (clayton.lewis@colorado.edu) is a professor in the Department of Computer Science at the University of Colorado at Boulder.

Michele H. Jackson (michele.jackson@colorado.edu) is an associate professor in the Department of Communication at the University of Colorado at Boulder.

William M. Waite (william.waite@colorado.edu) is a professor emeritus in the Department of Electrical, Computer, and Energy Engineering at the University of Colorado at Boulder.

Algorithmic solutions can help reduce energy consumption in computing environs.

BY SUSANNE ALBERS

Energy-Efficient Algorithms

ENERGY CONSERVATION IS a major concern today. Federal programs provide incentives to save energy and promote the use of renewable energy resources. Individuals, companies, and organizations seek energy-efficient products as the energy cost to run equipment has grown to be a major factor.

Energy consumption is also critical in computer systems, in terms of both cost and availability. Electricity costs impose a substantial strain on the budget of data and computing centers. Google engineers, maintaining thousands of servers, warned that if power consumption continues to grow, power costs can easily overtake hardware costs by a large margin.¹² In office environments, computers and monitors account for the highest energy consumption after lighting. Power dissipation is also a major concern in portable,

battery-operated devices that have proliferated rapidly in recent years. Each of us has experienced the event that the battery of our laptop or mobile phone is depleted. The issue is even more serious in autonomous, distributed devices such as sensor networks where the charging of batteries is difficult or impossible. Finally, energy dissipation causes thermal problems. Most of the energy consumed by a system is converted into heat, resulting in wear and reduced reliability of hardware components.

For these reasons, energy has become a leading design constraint for computing devices. Hardware engineers and system designers explore new directions to reduce the energy consumption of their products. The past years have also witnessed considerable research interest in algorithmic techniques to save energy. This survey reviews results that have been developed in the algorithms community to solve problems in energy management. For a given problem, the goal is to design *energy-efficient algorithms* that reduce energy consumption while minimizing compromise to service. An important aspect is that these algorithms must achieve a provably good performance.

This article focuses on the system and device level: How can we minimize energy consumption in a single computational device? We

» key insights

- **Energy management has become an important issue in computing environments. Algorithmic techniques provide effective solutions for energy savings, complementing hardware and systems-based approaches.**
- **Energy-efficient algorithms have been developed for a range of fundamental power management and dynamic speed-scaling problems that arise in many environments.**
- **Energy conservation involves decision making with incomplete information about the future. Energy-efficient algorithms achieve a provably good performance relative to the true optimum.**



ILLUSTRATION BY MARK KNOL

first study power-down mechanisms that conserve energy by transitioning a device into low-power standby or sleep modes. Then we address dynamic speed scaling in variable-speed processors. This relatively new technique saves energy by utilizing the full speed/frequency spectrum of a processor and applying low speeds whenever possible. Finally, we consider some optimization problems in wireless networks from an energy savings perspective.

We remark that all the above problems have also been studied in the systems literature. The corresponding papers also present algorithmic

approaches but usually do not prove performance guarantees.

Power-Down Mechanisms

Power-down mechanisms are well-known and widely used techniques to save energy. We encounter them on an everyday basis. The display of our desktop turns off after some period of inactivity. Our laptop transitions to a standby or hibernate mode if it has been idle for a while. In these settings, there usually exist idleness thresholds that specify the length of time after which a system is powered down when not in use. The following natural question arises: Is it possible

to design strategies that determine such thresholds and always achieve a provably good performance relative to the optimum solution? There exists a rich literature on power-down mechanisms, ranging from algorithmic to stochastic and learning-based approaches. This article concentrates on algorithmic solutions. We refer the reader to Benini et al. and Irani et al.^{14, 25} for surveys on other techniques.

Power Management and Competitiveness

Problem setting: In a general scenario, we are given a device that always resides in one of several states. In

addition to the active state there can be, for instance, standby, suspend, sleep, and full-off states. These states have individual power consumption rates. The energy incurred in transitioning the system from a higher-power to a lower-power state is usually negligible. However, a power-up operation consumes a significant amount of energy. Over time, the device experiences an alternating sequence of active and idle periods. During active periods, the system must reside in the active mode to perform the required tasks. During idle periods, the system may be moved to lower-power states. An algorithm has to decide when to perform the transitions and to which states to move. The goal is to minimize the total energy consumption. As the energy consumption during the active periods is fixed, assuming that prescribed tasks have to be performed, we concentrate on energy minimization in the idle intervals. In fact, we focus on any idle period and optimize the energy consumption in any such time window.

This power management problem is an *online problem*, that is, at any time a device is not aware of future events. More specifically, in an idle period, the system has no information when the period ends. Is it worthwhile to move to a lower-power state and benefit from the reduced energy consumption, given that the system must finally be powered up again at a cost to the active mode?

Performance analysis: Despite the handicap of not knowing the future, an online strategy should achieve a provably good performance. Here the algorithms community resorts to *competitive analysis*, where an online algorithm *ALG* is compared to an optimal offline algorithm *OPT*.³⁸ *OPT* is an omniscient strategy that knows the entire future and can compute a state transition schedule of minimum total energy. Online algorithm *ALG* is called *c-competitive* if for every input, such as, for any idle period, the total energy consumption of *ALG* is at most *c* times that of *OPT*.

Competitive analysis provides a strong worst-case performance guarantee. An online strategy must perform well on all inputs (idle periods) that might even be generated by an



Energy has become a leading design constraint for computing devices. Hardware engineers and system designers explore new directions to reduce energy consumption of their products.



adversary. This adversarial scenario may seem pessimistic but it is consistent with classical algorithm analysis that evaluates strategies in terms of their worst-case resources, typically running time or memory requirements. In this section, we will mostly study algorithms using competitive analysis but will also consider performance on inputs that are generated according to probability distributions.

In the following, we will first study systems that consist of two states only. Then we will address systems with multiple states. We stress that we consider the minimization of energy. We ignore the delay that arises when a system is transitioned from a lower-power to a higher-power state.

Systems with Two States

Consider a two-state system that may reside in an active state or in a sleep state. Let r be the power consumption rate, measured in energy units per time unit, in the active state. The power consumption rate in the sleep mode is assumed to be 0. The results we present in the following generalize to an arbitrary consumption rate in the sleep mode. Let β energy units, where $\beta > 0$, be required to transition the system from the sleep state to the active state. We assume that the energy of transitioning from the active to the sleep state is 0. If this is not the case, we can simply fold the corresponding energy into the cost β incurred in the next power-up operation. The system experiences an idle period whose length T is initially unknown.

An optimal offline algorithm *OPT*, knowing T in advance, is simple to formulate. We compare the value of rT , which is the total energy consumed during the idle period when residing in the active mode, to the power-up cost of β . If $rT < \beta$, *OPT* remains in the active state throughout the idle period as transitioning between the active and sleep modes costs more. If $rT \geq \beta$, using the sleep mode is beneficial. In this case *OPT* transitions to the sleep state right at the beginning of the idle period and powers up to the active state at the end of the period.

The following deterministic online algorithm mimics the behavior of *OPT*, which uses the sleep mode on idle periods of length at least β/r .

Algorithm ALG-D: In an idle period first remain in the active state. After β/r time units, if the period has not ended yet, transition to the sleep state.

It is easy to prove that *ALG-D* is 2-competitive. We only need to consider two cases. If $rT < \beta$, then *ALG-D* consumes rT units of energy during the idle interval and this is in fact equal to the consumption of *OPT*. If $rT \geq \beta$, then *ALG-D* first consumes $r \cdot \beta/r = \beta$ energy units to remain in the active state. An additional power-up cost of β is incurred at the end of the idle interval. Hence, *ALG-D*'s total cost is 2β , while *OPT* incurs a cost of β for the power-up operation at the end of the idle period.

It is also easy to verify that no deterministic online algorithm can achieve a competitive ratio smaller than 2. If an algorithm transitions to the sleep state after exactly t time units, then in an idle period of length t it incurs a cost of $tr + \beta$ while *OPT* pays $\min\{rt, \beta\}$ only.

We remark that power management in two-state systems corresponds to the famous ski-rental problem, a cornerstone problem in the theory of online algorithms, see, for example, Irani and Karlin.²⁶

Interestingly, it is possible to beat the competitiveness of 2 using randomization. A randomized algorithm transitions to the sleep state according to a probability density function $p(t)$. The probability that the system powers down during the first t_0 time units of an idle period is $\int_0^{t_0} p(t)dt$. Karlin et al.²⁸ determined the best probability distribution. The density function is the exponential function $e^{rt/\beta}$, multiplied by the factor $\frac{r}{(e-1)\beta}$ to ensure that $p(t)$ integrated over the entire time horizon is 1, that is, the system is definitely powered down at some point.

Algorithm ALG-R: Transition to the sleep state according to the probability density function

$$p(t) = \begin{cases} \frac{r}{(e-1)\beta} e^{rt/\beta} & 0 \leq t \leq \beta/r \\ 0 & \text{otherwise.} \end{cases}$$

ALG-R achieves a considerably improved competitiveness, as compared to deterministic strategies. Results by Karlin et al.²⁸ imply that *ALG-R* attains a competitive ratio of $\frac{e}{e-1} \approx 1.58$, where $e \approx 2.71$ is the Eulerian

number. More precisely, in any idle period the expected energy consumption of *ALG-R* is not more than $\frac{e}{e-1}$ times that of *OPT*. Again, $\frac{e}{e-1}$ is the best competitive ratio a randomized strategy can obtain.²⁸

From a practical point of view, it is also instructive to study stochastic settings where the length of idle periods is governed by probability distributions. In practice, short periods might occur more frequently. Probability distributions can also model specific situations where either very short or very long idle periods are more likely to occur, compared to periods of medium length. Of course, such a probability distribution may not be known in advance but can be learned over time. In the following, we assume that the distribution is known.

Let $Q = (q(T))_{0 \leq T < \infty}$ be a fixed probability distribution on the length T of idle periods. For any $t \geq 0$, consider the deterministic algorithm *ALG_t* that always powers down after exactly t time units. If the idle period ends before *ALG_t* powers down, such as, if $T < t$, then the algorithm remains in the active state for the duration of the idle interval and uses an energy of rT . If the idle period has not yet ended when *ALG_t* powers down, such as, if $T \geq t$, then the algorithm incurs a fixed energy of $rt + \beta$ because an energy of rt is consumed before the system is powered down and a cost β is incurred to transition back to the active mode. In order to determine the expected cost of *ALG_t*, we have to integrate over all possibilities for the length T of the idle period using the probability distribution Q . The two terms in the expression below represent the two cases. Note that the probability that the idle period has not yet ended when *ALG_t* powers down is $\int_t^\infty q(T)dT$.

$$E[ALG_t(Q)] = \int_0^t rTq(T)dT + (rt + \beta) \int_t^\infty q(T)dT \quad (1)$$

Karlin et al.²⁸ proposed the following strategy that, given Q , simply uses the best algorithm *ALG_t*.

Algorithm ALG-P: Given a fixed Q , let *A_Q* be the deterministic algorithm *ALG_t* that minimizes Equation 1.

Karlin et al. proved that for any Q , the expected energy consumption of

ALG-P is at most $\frac{e}{e-1}$ times the expected optimum consumption.

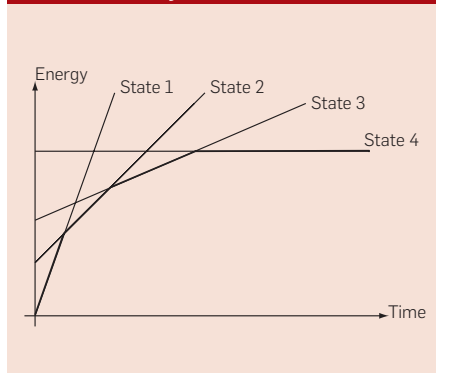
Systems with Multiple States

Many modern devices, beside the active state, have not only one but several low-power states. Specifications of such systems are given, for instance, in the Advanced Configuration and Power Management Interface (ACPI) that establishes industry-standard interfaces enabling power management and thermal management of mobile, desktop and server platforms. A description of the ACPI power management architecture built into Microsoft Windows operating systems can be found at <http://www.microsoft.com/whdc/system/pnppwr/powermgmt/default.mspx>.¹

Consider a system with ℓ states $s_1, \dots, s_\ell$. Let r_i be the power consumption rate of s_i . We number the states in order of decreasing rates, such as, $r_1 > \dots > r_\ell$. Hence s_1 is the active state and s_ℓ represents the state with lowest energy consumption. Let β_i be the energy required to transition the system from s_i to the active state s_1 . As transitions from lower-power states are more expensive we have $\beta_1 \leq \dots \leq \beta_\ell$. Moreover, obviously, $\beta_1 = 0$. We assume again that transitions from higher-power to lower-power states incur 0 cost because the corresponding energy is usually negligible. The goal is to construct a state transition schedule minimizing the total energy consumption in an idle period.

Irani et al.²⁴ presented online and offline algorithms. They assume that the transition energies are *additive*, such as, transitioning from a lower-power state s_j to a higher-power state s_i , where $i < j$, incurs a cost of $\beta_j - \beta_i$. An

Figure 1. Illustration of the optimum cost in a four-state system.



optimal off-line strategy *OPT*, knowing the length T of the idle period, is simple to state. We first observe that *OPT* changes states only at the beginning and at the end of the idle period. No transitions are performed during the period: Moving from a higher-power state s_i to a lower-power state s_j , with $j > i$ during the period is not sensible as s_i consumes more energy and a power-up cost of at least β_j must be paid anyway to reach the active mode at the end of the period. A better option is to immediately use s_j . Similarly, a transition from a lower-power state s_j to a higher-power state s_i does not make sense as s_j consumes less energy and the transition cost of $\beta_j - \beta_i$ can better be spent later when transitioning back to the active mode. If *OPT* uses state s_i throughout the idle period, its total energy consumption is $r_i T + \beta_i$. *OPT* simply chooses the state that minimizes the cost, that is, the optimum energy consumption is given by

$$OPT(T) = \min_{1 \leq i \leq \ell} \{r_i T + \beta_i\}.$$

Interestingly, for variable T the optimal cost has a simple graphical representation, see Figure 1. If we consider all linear functions $f_i(t) = r_i t + \beta_i$, representing the total energy consumption using state s_i , then the optimum energy consumption is given by the lower-envelope of the arrangement of lines.

One can use this lower-envelope to guide an online algorithm to select which state to use at any time. Let $S_{OPT}(t)$ denote the state used by *OPT* in an idle period of total length t , such as, $S_{OPT}(t)$ is the state $\arg \min_{1 \leq i \leq \ell} \{r_i t + \beta_i\}$. Irani et al.²⁴ proposed an algorithm called *Lower-Envelope* that traverses the state sequence as suggested by the optimum offline algorithm. At any time t in an idle period, the algorithm uses the state *OPT* would use if the period had a total length of t .

Algorithm Lower-Envelope: In an idle period, at any time t , use state $S_{OPT}(t)$.

Intuitively, over time, *Lower-Envelope* visits the states represented by the lower-envelope of the functions $f_i(t)$. If currently in state s_{i-1} , the strategy transitions to the next state s_i at time t_i , where t_i is the point in time when *OPT* starts favoring s_i over s_{i-1} . Formally t_i is the intersection of the lines $f_{i-1}(t)$ and $f_i(t)$, that is, the solution to the equation

$r_{i-1}t + \beta_{i-1} = r_i t + \beta_i$. Here we assume that states whose functions do not occur on the lower-envelope, at any time, are discarded. We remark that the algorithm is a generalization of *ALG-D* for two-state systems. Irani et al.²⁴ proved that *Lower-Envelope* is 2-competitive. This is the best competitiveness a deterministic algorithm can achieve in arbitrary state systems.

Irani et al.²⁴ also studied the setting where the length of idle periods is generated by a probability distribution $Q = (q(T))_{0 \leq T < \infty}$. They determine the time t_i when an online strategy should move from state s_{i-1} to s_i , $2 \leq i \leq \ell$. To this end consider the deterministic online algorithm ALG_t that transitions to the lower-power state after exactly t time units. We determine the expected cost of ALG_t in an idle period whose length T is generated according to Q , assuming that only states s_{i-1} and s_i are available. Initially ALG_t resides in state s_{i-1} . If the idle period ends before ALG_t transitions to the lower-power state, such as, if $T < t$, then the energy consumption is $r_{i-1}T$. If the idle period has not ended yet when ALG_t transitions to the lower-power state, such as, if $T \geq t$, the algorithm incurs an energy $r_{i-1}t$ while residing in s_{i-1} during the first t time units and an additional energy of $r_i(T - t)$ when in state s_i during the remaining $T - t$ time units. At the end of the idle period, a power-up cost of $\beta_i - \beta_{i-1}$ is paid to transition from s_i back to s_{i-1} . Hence, in this case ALG_t incurs a total energy of $r_{i-1}t + (T - t)r_i + \beta_i - \beta_{i-1}$. The expected cost of ALG_t , assuming that only s_{i-1} and s_i are available, is

$$\int_0^t r_{i-1} T q(T) dT + \int_t^\infty (r_{i-1}t + (T - t)r_i + \beta_i - \beta_{i-1}) q(T) dT.$$

Let t_i be the time t that minimizes the above expression. Irani et al.²⁴ proposed the following algorithm.

Algorithm ALG-P(ℓ): Change states at the transition times $t_2, \dots, t_\ell$ defined above.

ALG-P(ℓ) is a generalization of *ALG-P* for two-state systems. Irani et al. proved that for any fixed probability distribution Q , the expected energy consumption of *ALG-P(ℓ)* is no more than $\frac{e}{e-1}$ times the expected optimum consumption. Furthermore, Irani et

al. presented an approach for learning an initially unknown Q . They combined the approach with *ALG-P(ℓ)* and performed experimental tests for an IBM mobile hard drive with four power states. It shows that the combined scheme achieves low energy consumptions close to the optimum and usually outperforms many single-value prediction algorithms.

Augustine et al.⁵ investigate generalized multistate systems in which the state transition energies may take arbitrary values. Let $\beta_{ij} \geq 0$ be the energy required to transition from s_i to s_j , $1 \leq i, j \leq \ell$. Augustine et al. demonstrate that *Lower-Envelope* can be generalized and achieves a competitiveness of $3 + 2\sqrt{2} \approx 5.8$. This ratio holds for any state system. Better bounds are possible for specific systems. Augustine et al. devise a strategy that, for a given system S , achieves a competitive ratio arbitrarily close to the best competitiveness c^* possible for S . Finally, the authors consider stochastic settings and develop optimal state transition times.

Dynamic Speed Scaling

Many modern microprocessors can run at variable speed. Examples are the Intel SpeedStep and the AMD processor PowerNow. High speeds result in higher performance but also high energy consumption. Lower speeds save energy but performance degrades. The well-known cube-root rule for CMOS devices states that the speed s of a device is proportional to the cube-root of the power or, equivalently, the power is proportional to s^3 . The algorithms literature considers a generalization of this rule. If a processor runs at speed s , then the required power is s^α , where $\alpha > 1$ is a constant. Obviously, energy consumption is power integrated over time. The goal is to dynamically set the speed of a processor so as to minimize energy consumption, while still providing a desired quality of service.

Dynamic speed scaling leads to many challenging scheduling problems. At any time a scheduler has to decide not only which job to execute but also which speed to use. Consequently, there has been considerable research interest in the design and analysis of efficient scheduling algorithms. This section reviews the most important results developed over the past years.

We first address scheduling problems with hard job deadlines. Then we consider the minimization of response times and other objectives.

In general, two scenarios are of interest. In the offline setting, all jobs to be processed are known in advance. In the online setting, jobs arrive over time, and an algorithm, at any time, has to make scheduling decisions without knowledge of any future jobs. Online strategies are evaluated again using competitive analysis. Online algorithm *ALG* is c -competitive if, for every input, the objective function value (typically the energy consumption) of *ALG* is within c times the value of an optimal solution.

Scheduling with Deadlines

In a seminal paper, initiating the algorithmic study of speed scaling, Yao et al.⁴⁰ investigated a scheduling problem with strict job deadlines. At this point, this framework is by far the most extensively studied algorithmic speed scaling problem.

Consider n jobs $J_1, \dots, J_n$ that have to be processed on a variable-speed processor. Each job J_i is specified by a release time r_i , a deadline d_i , and a processing volume w_i . The release time and the deadline mark the time interval in which the job must be executed. The processing volume is the amount of work that must be done to complete the job. Intuitively, the processing volume can be viewed as the number of CPU cycles necessary to finish the job. The processing time of a job depends on the speed. If J_i is executed at constant speed s , it takes w_i/s time units to

complete the job. Preemption of jobs is allowed, that is, the processing of a job may be suspended and resumed later. The goal is to construct a feasible schedule minimizing the total energy consumption.

The framework by Yao et al. assumes there is no upper bound on the maximum processor speed. Hence there always exists a feasible schedule satisfying all job deadlines. Furthermore, it is assumed that a continuous spectrum of speeds is available. We will discuss later how to relax these assumptions.

Fundamental algorithms: Yao et al.⁴⁰ first study the offline setting and develop an algorithm for computing optimal solutions, minimizing total energy consumption. The strategy is known as *YDS* referring to the initials of the authors. The algorithm proceeds in a series of iterations. In each iteration, a time interval of maximum density is identified and a corresponding partial schedule is constructed. Loosely speaking, the density of an interval I is the minimum average speed necessary to complete all jobs that must be scheduled in I . A high density requires a high speed. Formally, the density Δ_I of a time interval $I = [t, t']$ is the total work to be completed in I divided by the length of I . More precisely, let S_I be the set of jobs J_i that must be processed in I because their release time and deadline are in I , such as, $[r_i, d_i] \subseteq I$. The corresponding total processing volume is $\sum_{J_i \in S_I} w_i$. Then

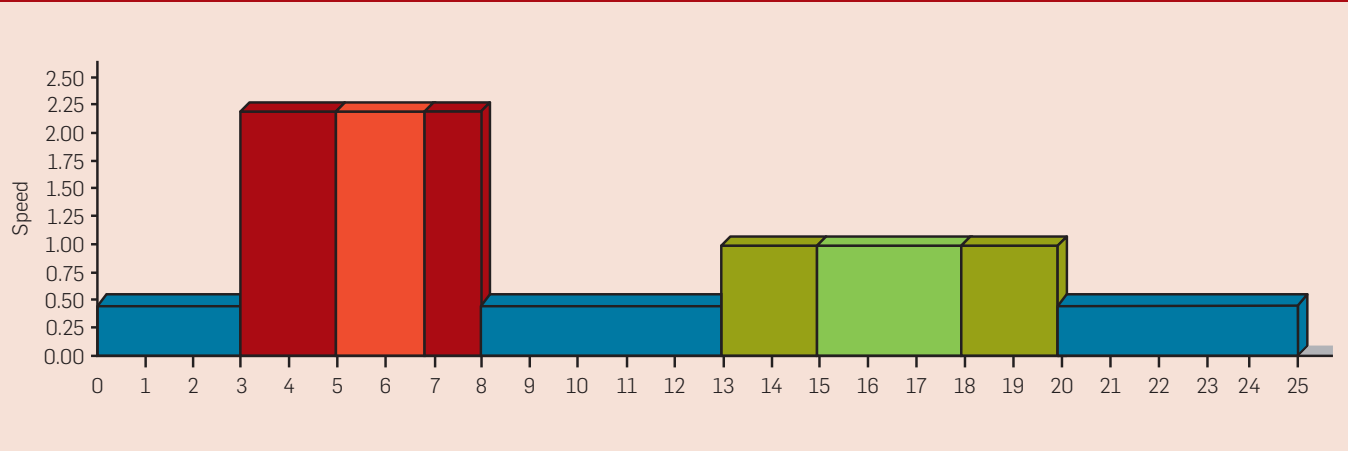
$$\Delta_I = \frac{1}{|I|} \sum_{J_i \in S_I} w_i.$$

Algorithm *YDS* repeatedly determines the interval I of maximum density. In such an interval I , the algorithm schedules the jobs of S_I at speed Δ_I using the *Earliest Deadline First (EDF)* policy. This well-known policy always executes the job having the earliest deadline, among the available unfinished jobs. After this assignment, *YDS* removes set S_I as well as time interval I from the problem instance. More specifically, for any unscheduled job J_i whose deadline is in the interval I , the new deadline is set to the beginning of I because the time window I is not available anymore for the processing of J_i . Formally, for any J_i with $d_i \in I$, the new deadline time is set to $d_i := t$. Similarly, for any unscheduled J_i whose release time is in I , the new release time is set to the end of I . Again, formally for any J_i with $r_i \in I$, the new release time is $r_i := t'$. Time interval I is discarded. This process repeats until there are no more unscheduled jobs. We give a summary of the algorithm in pseudocode.

Algorithm *YDS*: Initially $\mathcal{J} := \{J_1, \dots, J_n\}$. While $\mathcal{J} \neq \emptyset$, execute the following two steps. (1) Determine the interval I of maximum density. In I , process the jobs of S_I at speed Δ_I according to *EDF*. (2) Set $\mathcal{J} := \mathcal{J} \setminus S_I$. Remove I from the time horizon and update the release times and deadlines of unscheduled jobs accordingly.

Figure 2 depicts the schedule constructed by *YDS* on an input instance with five jobs. Jobs are represented by colored rectangles, each job having a different color. The rectangle heights correspond to the speeds at which the jobs are processed. Time is drawn on

Figure 2. An input instance with five jobs specified as $J_i = (r_i, d_i, w_i)$. Blue $J_1 = (0, 25, 9)$; red $J_2 = (3, 8, 7)$; orange $J_3 = (5, 7, 4)$; dark green $J_4 = (13, 20, 4)$; light green $J_5 = (15, 18, 3)$.



the horizontal axis. In the first iteration YDS identifies $I_1 = [3, 8]$ as interval of maximum density, along with set $S_{I_1} = \{J_2, J_3\}$. In I_1 , the red job J_2 is preempted at time 5 to give preference to the orange job J_3 having an earlier deadline. In the second iteration $I_2 = [13, 20]$ is the maximum density interval. The dark green and light green jobs are scheduled; preemption is again used once. In the third iteration, the remaining job J_3 is scheduled in the available time slots.

Obviously, when identifying intervals of maximum density, YDS only has to consider intervals whose boundaries are equal to the release times and deadlines of the jobs. A straightforward implementation of the algorithm has a running time of $O(n^3)$. Li et al.³⁴ showed that the time can be reduced to $O(n^2 \log n)$. Further improvements are possible if the job execution intervals form a tree structure.³³

Yao et al.⁴⁰ also devised two elegant online algorithms, called *Average Rate* and *Optimal Available*. Whenever a new job J_i arrives at time r_i , its deadline d_i and processing volume w_i are known. For any incoming job J_j , *Average Rate* considers the density $\delta_i = w_i/(d_i - r_i)$, which is the minimum average speed necessary to complete the job in time if no other jobs were present. At any time t , the speed $s(t)$ is set to the accumulated density of jobs *active* at time t . A job J_i is *active* at time t if it is available for processing at that time, such as, if $t \in [r_i, d_i]$. Available jobs are scheduled according to the *EDF* policy.

Algorithm Average Rate: At any time t , use a speed of $s(t) = \sum_{J_i: t \in [r_i, d_i]} \delta_i$. Available unfinished jobs are scheduled using *EDF*.

Yao et al.⁴⁰ analyzed *Average Rate* and proved that the competitive ratio is upper bounded by $2^{\alpha-1}\alpha^\alpha$, for any $\alpha \geq 2$. Bansal et al.⁶ showed that the analysis is essentially tight by providing a nearly matching lower bound.

The second strategy *Optimal Available* is computationally more expensive than *Average Rate*. It always computes an optimal schedule for the currently available workload. A recomputation is necessary whenever a new job arrives. A new optimal schedule for the future time horizon can be constructed using YDS .

Algorithm Optimal Available: Whenever a new job arrives, compute an

optimal schedule for the currently available unfinished jobs.

Bansal et al.⁹ gave a comprehensive analysis of the above algorithm and proved that the competitive ratio is exactly α^α . Hence, in terms of competitiveness, *Optimal Available* is better than *Average Rate*. Bansal et al.⁹ also presented a new online algorithm, called *BKP* according to the initials of the authors, which can be viewed as approximating the optimal speeds of YDS in an online manner. Again, the algorithm considers interval densities. For times t, t_1 , and t_2 with $t_1 < t \leq t_2$, let $w(t, t_1, t_2)$ be the total processing volume of jobs that have arrived by time t , have a release time of at least t_1 and a deadline of at most t_2 . Then, intuitively, $\max_{t_1, t_2} w(t, t_1, t_2)/(t_2 - t_1)$ is an estimate of the speed used by YDS , based on the knowledge of jobs that have arrived by time t . The new algorithm *BKP* approximates this speed by considering specific time windows $[et - (e-1)t', t']$, for $t' > t$, of length $e(t' - t)$. The corresponding necessary speed is then multiplied by a factor of e .

Algorithm BKP: At any time t use a speed of $e \cdot s(t)$, where

$$s(t) = \max_{t' > t} \frac{w(t, et - (e-1)t', t')}{e(t' - t)}.$$

Available unfinished jobs are processed using *EDF*.

Bansal et al.⁹ proved that *BKP* achieves a competitive ratio of $2(\frac{\alpha}{\alpha-1})^\alpha e^\alpha$, which is better than the competitiveness of *Optimal Available* for large values of α .

All the above online algorithms attain constant competitive ratios that depend on α and no other problem parameter. The dependence on α is exponential. For small values of α , which occur in practice, the competitive ratios are reasonably small. A result by Bansal et al.⁹ implies that the exponential dependence on α is inherent to the problem. Any randomized online algorithm has a competitiveness of at least $\Omega((4/3)^\alpha)$.

Refinements—Bounded speed: The problem setting considered so far assumes a continuous, unbounded spectrum of speeds. However, in practice only a finite set of discrete speed levels $s_1 < s_2 < \dots < s_d$ is available. The

offline algorithm YDS can be adapted easily to handle feasible job instances, such as, inputs for which feasible schedules exist using the restricted set of speeds. Note that feasibility can be checked easily by always using the maximum speed s_d and scheduling available jobs according to the *EDF* policy. Given a feasible job instance, the modification of YDS is as follows. We first construct the schedule according to YDS . For each identified interval I of maximum density, we approximate the desired speed Δ_I by the two adjacent speed levels s_k and s_{k+1} , such that $s_k < \Delta_I < s_{k+1}$. Speed s_{k+1} is used first for some δ time units and s_k is used for the last $|I| - \delta$ time units in I , where δ is chosen such that the total work completed in I is equal to the original amount of $|I|\Delta_I$. An algorithm with an improved running time of $O(dn \log n)$ was presented by Li and Yao.³⁵

If the given job instance is not feasible, the situation is more delicate. In this case it is impossible to complete all the jobs. The goal is to design algorithms that achieve good *throughput*, which is the total processing volume of jobs finished by their deadline, and at the same time optimize energy consumption. Papers^{7, 17} present algorithms that even work online. At any time the strategies maintain a pool of jobs they intend to complete. Newly arriving jobs may be admitted to this pool. If the pool contains too large a processing volume, jobs are expelled such that the throughput is not diminished significantly. The algorithm by Bansal et al.⁷ is 4-competitive in terms of throughput and constant competitive with respect to energy consumption.

Temperature minimization: High processor speeds lead to high temperatures, which impair a processor's reliability and lifetime. Bansal et al.⁹ consider the minimization of the maximum temperature that arises during processing. They assume that cooling follows Newton's Law, which states that the rate of cooling of a body is proportional to the difference in temperature between the body and the environment. Bansal et al.⁹ show that algorithms YDS and *BKP* have favorable properties. For any jobs sequence, the maximum temperature is within a constant factor of the minimum possible

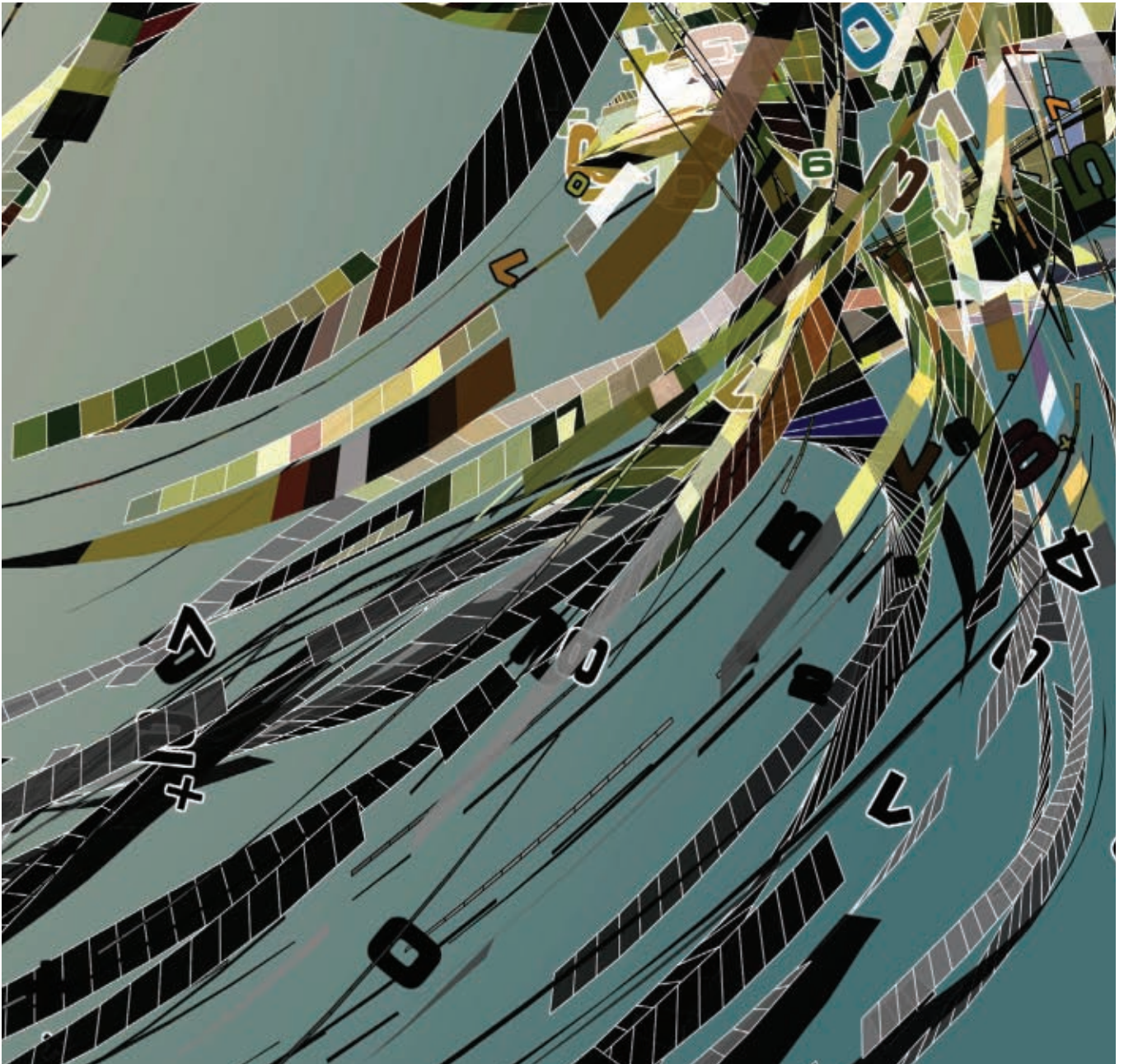


ILLUSTRATION BY MARK KNOL

maximum temperature, for any cooling parameter a device may have.

Sleep states: Irani et al.²³ investigate an extended problem setting where a variable-speed processor may be transitioned into a sleep state. In the sleep state, the energy consumption is 0 while in the active state even at speed 0 some non-negative amount of energy is consumed. Hence, Irani et al.²³ combine speed scaling with power-down mechanisms. In the standard setting without sleep state, algorithms tend to use low speed levels subject to release time and deadline constraints. In contrast, in the setting with sleep state it can be beneficial to speed up a job

so as to generate idle times in which the processor can be transitioned to the sleep mode. Irani et al.²³ develop online and offline algorithms for this extended setting. Baptiste et al.¹¹ and Demaine et al.²¹ also study scheduling problems where a processor may be set asleep, albeit in a setting without speed scaling.

Minimizing Response Time

A classical objective in scheduling is the minimization of response times. A user releasing a task to a system would like to receive feedback, say the result of a computation, as quickly as possible. User satisfaction often

depends on how fast a device reacts. Unfortunately, response time minimization and energy minimization are contradicting objectives. To achieve fast response times, a system must usually use high processor speeds, which lead to high energy consumption. On the other hand, to save energy, low speeds should be used, which result in high response times. Hence, one has to find ways to integrate both objectives.

Consider n jobs $J_1, \dots, J_n$ that have to be scheduled on a variable-speed processor. Each job J_i is specified by a release time r_i and a processing volume w_i . When a job arrives, its processing volume is known. Preemption of

jobs is allowed. In the scheduling literature, response time is referred to as *flow time*. The flow time f_i of a job J_i is the length of the time interval between release time and completion time of the job. We seek schedules minimizing the total flow time $\sum_{i=1}^n f_i$.

Limited energy: Pruhs et al.³⁷ study a problem setting where a fixed energy volume E is given and the goal is to minimize the total flow time of the jobs. The authors assume all jobs have the same processing volume. By scaling, we can assume all jobs have unit-size. Pruhs et al.³⁷ consider the offline scenario where all the jobs are known in advance and show that optimal schedules can be computed in polynomial time. However, in this framework with a limited energy volume it is difficult to construct good online algorithms. If future jobs are unknown, it is unclear how much energy to invest for the currently available tasks.

Energy plus flow times: Albers and Fujiwara² proposed another approach to integrate energy and flow time minimization. They consider a combined objective function that simply adds the two costs. Let E denote the energy consumption of a schedule. We wish to minimize $g = E + \sum_{i=1}^n f_i$. By multiplying either the energy or the flow time by a scalar, we can also consider a weighted combination of the two costs, expressing the relative value of the two terms in the total cost. Albers and Fujiwara concentrate on unit-size jobs and show that optimal offline schedules can be constructed in polynomial time using a dynamic programming approach. In fact the algorithm can also be used to minimize the total flow time of jobs given a fixed energy volume.

Most of the work by the authors² is concerned with the online setting where jobs arrive over time. Albers and Fujiwara present a simple online strategy that processes jobs in batches and achieves a constant competitive ratio. Batched processing allows one to make scheduling decisions, which are computationally expensive, only every once in a while. This is certainly an advantage in low-power computing environments. Nonetheless, Albers and Fujiwara conjectured that the following algorithm achieves a better performance with respect to the minimization of g : at any time, if there are ℓ

active jobs, use speed $\sqrt[\ell]{\ell}$. A job is active if it has been released but is still unfinished. Intuitively, this is a reasonable strategy because, in each time unit, the incurred energy of $(\sqrt[\ell]{\ell})^\ell = \ell$ is equal to the additional flow time accumulated by the ℓ jobs during that time unit. Hence, both energy and flow time contribute the same value to the objective function. The algorithm and variants thereof have been the subject of extensive analyses,^{7, 8, 10, 32} not only for unit-size jobs but also for arbitrary size jobs. Moreover, unweighted and weighted flow times have been considered.

The currently best result is due to Bansal et al.⁸ They modify the above algorithm slightly by using a speed of $\sqrt[\ell]{\ell} + 1$ whenever ℓ jobs are active. Inspired by a paper of Lam et al.,³² they apply the *Shortest Remaining Processing Time (SRPT)* policy to the available jobs. More precisely, at any time among the active jobs, the one with the least remaining work is scheduled.

Algorithm Job Count: At any time if there are $\ell \geq 1$ active jobs, use speed $\sqrt[\ell]{\ell} + 1$. If no job is available, use speed 0. Always schedule the job with the least remaining unfinished work.

Bansal et al.⁸ proved that *Job Count* is 3-competitive for arbitrary size jobs. Further work considering the weighted flow time in objective function g can be found in Bansal et al.^{8, 10} Moreover, Bansal et al. and Lam et al.^{7, 32} propose algorithms for the setting that there is an upper bound on the maximum processor speed.

All the above results assume that when a job arrives, its processing volume is known. Papers^{18, 32} investigate the harder case that this information is not available.

Extensions and Other Objectives

Parallel processors: The results presented so far address single-processor architectures. However, energy consumption is also a major concern in multiprocessor environments. Currently, relatively few results are known. Albers et al.³ investigate deadline-based scheduling on m identical parallel processors. The goal is to minimize the total energy on all the machines. The authors first settle the complexity of the offline problem by showing that computing optimal schedules is NP-hard, even for

unit-size jobs. Hence, unless $P = NP$, optimal solutions cannot be computed efficiently. Albers et al.³ then develop polynomial time offline algorithms that achieve constant factor approximations, such as, for any input the consumed energy is within a constant factor of the true optimum. They also devise online algorithms attaining constant competitive ratios. Lam et al.³⁰ study deadline-based scheduling on two speed-bounded processors. They present a strategy that is constant competitive in terms of throughput maximization and energy minimization.

Bunde¹⁵ investigates flow time minimization in multiprocessor environments, given a fixed energy volume. He presents hardness results as well as approximation guarantees for unit-size jobs. Lam et al.³¹ consider the objective function of minimizing energy plus flow times. They design online algorithms achieving constant competitive ratios.

Makespan minimization: Another basic objective function in scheduling is makespan minimization, that is, the minimization of the point in time when the entire schedule ends. Bunde¹⁵ assumes that jobs arrive over time and develops algorithms for single- and multiprocessor environments. Pruhs et al.³⁶ consider tasks having precedence constraints defined between them. They devise algorithms for parallel processors given a fixed energy volume.

Wireless Networks

Wireless networks such as ad hoc networks and sensor networks have received considerable attention over the last few years. Prominent applications of such networks are habitat observation, environmental monitoring, and forecasting. Network nodes usually have very limited battery capacity so that effective energy management strategies are essential to improve the lifetime of a network. In this survey, we focus on two algorithmic problems that have received considerable interest in the research community recently. Moreover, these problems can be viewed as scheduling problems and hence are related to the topics addressed in the previous sections.

Network Topologies

Wireless ad hoc networks do not have a fixed infrastructure. The network basically consists of a collection of radio stations with antennas for sending and receiving signals. During transmission a station s has to choose a transmission power P_s , taking into account that the signal strength decreases over distance. The signal is successfully received by a station t only if $P_s/\text{dist}(s,t)^\alpha > \gamma$. Here $\text{dist}(s,t)$ denotes the distance between s and t , coefficient $\alpha > 1$ is the attenuation rate and $\gamma > 0$ is a transmission quality parameter. In practice the attenuation rate is in the range between 2 and 5. Without loss of generality we may assume $\gamma = 1$.

In data transmission, a very basic operation is *broadcast*, where a given source node wishes to send a piece of information to all other nodes in the network. We study the problem of designing *broadcast topologies* allowing energy-efficient broadcast operations in wireless networks. Consider a set V of n nodes that are located in the real plane $\mathbb{R}^2$. A source node $s \in V$ has to disseminate a message to all other nodes in the network. However s does not have to inform all $v \in V$ directly. Instead nodes may serve as relay stations. If v receives the message and transmits it to $w_1, \dots, w_k$, then v has to use a power of $P_v = \max_{1 \leq j \leq k} \text{dist}(v, w_j)^\alpha$. The goal is to find a topology, that is, a transmission schedule that minimizes the total power/energy $E = \sum_{v \in V} P_v$ incurred by all the nodes. Note that such a schedule corresponds to a tree T that is rooted at s and contains all the nodes of V . The children of a node v are the nodes to which v transfers the message.

Clementi et al.¹⁹ showed that the computation of optimal schedules is NP-hard. Therefore one resorts to approximations. An algorithm *ALG* achieves a c -approximation if for every input, such as, for every node set V , the solution computed by *ALG* incurs an energy consumption of no more than c times the optimum value. Wan et al.³⁹ investigate various algorithms in terms of their approximation guarantees. The most extensively studied strategy is *MST*. For a given node set V , *MST* computes a standard minimum spanning tree T , such as, a tree of minimum total edge length containing all the vertices



We need a better understanding of the speed-scaling techniques in multiprocessor environments as multicore architectures become more common not only in servers but in desktops and laptops.



of V (see, e.g., Cormen et al.²⁰). The tree is rooted at source node s . Data transmission is performed along the edges of T , that is, each node transmits a received message to all of its children in the tree. Intuitively, this algorithm is sensible because the small total edge length of a minimum spanning tree should lead to a small overall energy consumption.

Algorithm MST: For a given V , compute a minimum spanning tree rooted at s . Any node v transmits a given message to all of its children in T .

Many papers have analyzed *MST*, see Ambühl,⁴ Caragiannis et al.,¹⁶ Clementi et al.,¹⁹ Flammini et al.,²² and Wan et al.³⁹ and references therein. Ambühl⁴ proved that *MST* achieves a 6-approximation. The analysis is tight because Wan et al.³⁹ showed that the ratio is not smaller than 6. A new improved algorithm was recently presented by Caragiannis et al.¹⁶

From a practical point of view, another strategy, called *Broadcast Incremental Power (BIP)*, is very interesting. This algorithm constructs a broadcast tree in a series of iterations, starting from an initial tree T_0 that only contains s . In any iteration i , a new tree T_i is obtained from T_{i-1} by computing the smallest additional power necessary at any node of T_{i-1} to include (at least) one additional node $v \notin T_{i-1}$. The new node and the corresponding edge are added to T_{i-1} . Results by Ambühl⁴ and Wan et al.³⁹ imply that the approximation ratio c of *BIP* satisfies $13/3 \leq c \leq 6$. It would be interesting to develop tight bounds for this algorithm.

Data Aggregation

As mentioned above, sensor networks are typically used to monitor an environment, measuring, e.g., temperature or a chemical value. The data has to be transferred to a designated sink node that may perform further actions. Becchetti et al.¹³ and Korteweg et al.²⁹ develop energy-efficient protocols for data aggregation.

Suppose the transmission topology is given by a tree T rooted at the sink s . Data gathered at a network node v is transmitted along the path from v to s in T . Network nodes have the ability to combine data. If two or more data packets simultaneously reside at a node v , then v may merge these packets into a

single one and transfer it to the parent node, in the direction of s . The energy incurred by a network node is proportional to the number of packets sent.

Becchetti et al.¹³ assume that data items arrive over time. Each item i is specified by the node v_i where the item arises, an arrival time r_i and a deadline d_i by which the data must reach the sink. The goal is to find a feasible transmission schedule minimizing the maximum energy required at any node. Becchetti et al. show that the offline problem is NP-hard and present a 2-approximation algorithm. They also develop distributed online algorithms for synchronous as well as asynchronous communication models. Korteweg et al.²⁹ study a problem variant where the data items do not have deadlines but should reach the sink with low latency. They present algorithms that simultaneously approximate energy consumption and latency, considering again various communication models.

Conclusion

In this survey, we have reviewed algorithmic solutions to save energy. Another survey on algorithmic problems in power management was written by Irani and Pruhs.²⁷ Over the past months a large number of papers have been published, and we expect that energy conservation from an algorithmic point of view will continue to be an active research topic. There are many directions for future research. With respect to power-down mechanisms, for instance, it would be interesting to design strategies that take into account the latency that arises when a system is transitioned from a sleep state to the active state. Additionally, we need a better understanding of speed scaling techniques in multiprocessor environments as multicore architectures become more and more common not only in servers but also in desktops and laptops. Moreover, optimization problems in networks deserve further algorithmic investigation. At this point it would be interesting to study energy-efficient point-to-point communication, complementing the existing work on broadcast and data-aggregation protocols. Last but not least, the algorithms presented so far have to be analyzed

in terms of their implementation and execution cost: how much extra energy is incurred in executing the algorithms in realistic environments. **C**

References

1. <http://www.microsoft.com/whdc/system/ppnpwr/powermgmt/default.mspx>
2. Albers, S., Fujiwara, H. Energy-efficient algorithms for flow time minimization. *ACM Trans. Algorithms* 3 (2007).
3. Albers, S., Müller, F., Schmelzer, S. Speed scaling on parallel processors. In *Proceedings of the 19th ACM Symposium on Parallelism in Algorithms and Architectures* (2007), 289–298.
4. Ambühl, C. An optimal bound for the MST algorithm to compute energy efficient broadcast trees in wireless networks. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming* (2005), Springer LNCS 3580, 1139–1150.
5. Augustine, J., Irani, S., Swamy, C. Optimal power-down strategies. *SIAM J. Comput.* 37 (2008), 1499–1516.
6. Bansal, N., Bunde, D.P., Chan, H.-L., Pruhs K. Average rate speed scaling. In *Proceedings of the 8th Latin American Symposium on Theoretical Informatics* (2008), Springer LNCS 4957, 240–251.
7. Bansal, N., Chan, H.-L., Lam, T.-W., Lee, K.-L. Scheduling for speed bounded processors. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming* (2008), Springer LNCS 5125, 409–420.
8. Bansal, N., Chan, H.-L., Pruhs, K. Speed scaling with an arbitrary power function. In *Proceedings of the 20th ACM-SIAM Symposium on Discrete Algorithms* (2009).
9. Bansal, N., Kimbrel, T., Pruhs, K. Speed scaling to manage energy and temperature. *J. ACM* 54 (2007).
10. Bansal, N., Pruhs, K., Stein, C. Speed scaling for weighted flow time. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms* (2007), 805–813.
11. Baptiste, P., Chrobak, M., Dürr C. Polynomial time algorithms for minimum energy scheduling. In *Proceedings of the 15th Annual European Symposium on Algorithms* (2007), Springer LNCS 4698, 136–150.
12. Barroso, L.A. The price of performance. *ACM Queue* 3 (2005).
13. Becchetti, L., Korteweg, P., Marchetti-Spaccamela, A., Skutella, M., Stougie, L., Vitaletti, A. Latency constrained aggregation in sensor networks. In *Proceedings of the 14th Annual European Symposium on Algorithms* (2006), Springer LNCS 4168, 88–99.
14. Benini, L., Bogliolo, A., De Micheli, G. A survey of design techniques for system-level dynamic power management. *IEEE Trans. VLSI Syst.* 8 (2000), 299–316.
15. Bunde, D.P. Power-aware scheduling for makespan and flow. In *Proceedings of the 18th Annual ACM Symposium on Parallel Algorithms and Architectures* (2006), 190–196.
16. Caragiannis, I., Flammini, M., Moscardelli, L. An exponential improvement on the MST heuristic for minimum energy broadcasting in ad hoc wireless networks. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming* (2007), Springer LNCS 4596, 447–458.
17. Chan, H.-L., Chan, W.-T., Lam, T.-W., Lee, K.-L., Mak, K.-S., Wong P.W.H. Energy efficient online deadline scheduling. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms* (2007), 795–804.
18. Chan, H.-L., Edmonds, J., Lam, T.-W., Lee, L.-K., Marchetti-Spaccamela, A., Pruhs, K. Nonclairvoyant speed scaling for flow and energy. In *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science* (2009), 255–264.
19. Clementi, A.E.F., Crescenzi, P., Penna, P., Rossi, G., Vocco, P. On the complexity of computing minimum energy consumption broadcast subgraphs. In *Proceedings of the 18th International Symposium on Theoretical Aspects of Computer Science* (2001), Springer 2010, 121–131.
20. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. *Introduction to Algorithms*, MIT Press and McGraw-Hill, 2001.
21. Demaine, E.D., Ghodsi, M., Hajiaghay, M.T., Sayedi-Roshkhar, A.S., Zadimoghaddam, M. Scheduling to minimize gaps and power consumption. In *Proceedings of the 19th Annual ACM Symposium on Parallel Algorithms and Architectures* (2007), 46–54.
22. Flammini, M., Klasing, R., Navarra, A., Perennes, S. Improved approximation results for the minimum energy broadcasting problem. *Algorithmica* 49 (2007), 318–336.
23. Irani, S., Shukla, S.K., Gupta, R. Algorithms for power savings. *ACM Trans. Algorithms* 3 (2007).
24. Irani, S., Shukla, S.K., Gupta, R.K. Online strategies for dynamic power management in systems with multiple power-saving states. *ACM Trans. Embedded Comput. Syst.* 2 (2003), 325–346.
25. Irani, S., Singh, G., Shukla, S.K., Gupta, R.K. An overview of the competitive and adversarial approaches to designing dynamic power management strategies. *IEEE Trans. VLSI Syst.* 13 (2005), 1349–1361.
26. Irani, S., Karlin, A.R. Online computation. In *Approximation Algorithms for NP-Hard Problems*. Hochbaum D. ed. PWS Publishing Company, 1997, 521–564.
27. Irani, S., Pruhs, K. Algorithmic problems in power management. *SIGACT News* 36 (2005), 63–76.
28. Karlin, A.R., Manasse, M.S., McGeoch, L.A., Owicki, S.S. Competitive randomized algorithms for nonuniform problems. *Algorithmica* 11 (1994), 542–571.
29. Korteweg, P., Marchetti-Spaccamela, A., Stougie, L., Vitaletti, A. Data aggregation in sensor networks: Balancing communication and delay costs. In *Proceedings of the 14th International Colloquium on Structural Information and Communication Complexity* (2007), Springer LNCS 4474, 139–150.
30. Lam, T.-W., Lee, L.-K., To, I.K.-K., Wong, P.W.H. Energy efficient deadline scheduling in two processor systems. In *Proceedings of the 18th International Symposium on Algorithms and Computation* (2007), Springer LNCS 4835, 476–487.
31. Lam, T.-W., Lee, L.-K., To, I.K.-K., Wong, P.W.H. Competitive non-migratory scheduling for flow time and energy. In *Proceedings of the 20th Annual ACM Symposium on Parallel Algorithms and Architectures* (2008), 256–264.
32. Lam, T.-W., Lee, L.-K., To, I.K.-K., Wong, P.W.H. Speed scaling functions for flow time scheduling based on active job count. In *Proceedings of the 16th Annual European Symposium on Algorithms* (2008), Springer LNCS 5193, 647–659.
33. Li, M., Liu, B.-J., Yao, F.F. Min-energy voltage allocation for tree-structured tasks. *J. Comb. Optim.* 11 (2006), 305–319.
34. Li, M., Yao, A.C., Yao, F.F. Discrete and continuous min-energy schedules for variable voltage processors. In *Proceedings of the National Academy of Sciences USA* 103 (2006), 3983–3987.
35. Li, M., Yao, F.F. An efficient algorithm for computing optimal discrete voltage schedules. *SIAM J. Comput.* 35 (2005), 658–671.
36. Pruhs, K., van Stee, R., Uthaisombut, P. Speed scaling of tasks with precedence constraints. *Theory Comput. Syst.* 43 (2008), 67–80.
37. Pruhs, K., Uthaisombut, P., Woeginger, G.J. Getting the best response for your erg. *ACM Trans. Algorithms* 4 (2008).
38. Sleator, D.D., Tarjan, R.E. Amortized efficiency of list update and paging rules. *Comm. ACM* 28 (1985), 202–208.
39. Wan, P.-J., Calinescu, G., Li, X.-Y., Frieder, O. Minimum-energy broadcasting in static ad hoc wireless networks. *Wireless Netw.* 8 (2002), 607–617.
40. Yao, F.F., Demers, A.J., Shenker, S. A scheduling model for reduced CPU energy. In *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science* (1995), 374–382.

Susanne Albers is a professor in the Department of Computer Science at Humboldt University, Berlin, Germany.

Work supported by a Gottfried Wilhelm Leibniz Award of the German Research Foundation.

research highlights

P. 98

**Technical
Perspective**
**Learning to Act
in Uncertain
Environments**

By Peter L. Bartlett

P. 99

Censored Exploration and the Dark Pool Problem

By Kuzman Ganchev, Yuriy Nevmyvaka, Michael Kearns,
and Jennifer Wortman Vaughan

P. 108

**Technical
Perspective**
**Automated Patching
Techniques:
The Fix Is In**

By Mark Harman

P. 109

Automatic Program Repair with Evolutionary Computation

By Westley Weimer, Stephanie Forrest, Claire Le Goues,
and ThanhVu Nguyen

Technical Perspective

Learning to Act in Uncertain Environments

By Peter L. Bartlett

THE PROBLEM OF decision making in an uncertain environment arises in many diverse contexts: deciding whether to keep a hard drive spinning in a netbook; choosing which advertisement to post to a Web site visitor; choosing how many newspapers to order so as to maximize profits; or choosing a route to recommend to a driver given limited and possibly out-of-date information about traffic conditions. All are sequential decision problems, since earlier decisions affect subsequent performance; all require adaptive approaches, since they involve significant uncertainty. The key issue in effectively solving problems like these is known as the *exploration/exploitation trade-off*: If I am at a crossroads, when should I go in the most advantageous direction among those that I have already explored, and when should I strike out in a new direction, in the hopes I will discover something better?

The following paper by Ganchev, Kearns, Nevmyvaka, and Vaughan considers a sequential decision problem from the financial domain: how to allocate stock orders across a variety of marketplaces, each with an unknown volume available, so as to maximize the number of orders that are filled. These marketplaces are known as *dark pools* because they allow traders to keep their transactions hidden. The popularity of these dark pools has grown enormously, as traders making large transactions hope to reduce their market impact, that is, the tendency for the price to move in an unfavorable direction. Because transactions are hidden, the characteristics of the various dark pools are uncertain, and can only be discovered by active exploration.

The broad approach followed by the authors is based on an intuitive heuristic that is reminiscent of a title you might encounter in the self-help section of a bookstore: “optimism in the

face of uncertainty.” The idea is to treat uncertain outcomes as optimistically as the data allows: pick the alternative that, in the best possible world, is consistent with our experiences so far, and leads to the best outcome. One alternative might be chosen either because it is clearly superior to all others or because there is not enough data to rule out that possibility. If it turns out this alternative is a poor choice, at least it leads to a reduction in uncertainty, so that in the future it can be confidently avoided. This approach naturally leads to a balance between the desire to exploit information that has already been gathered, and the need to explore uncertain alternatives.

The authors illustrate how the optimism heuristic can be successfully applied to the dark pools problem. One striking feature of this result is that their approach is successful despite

The following paper considers a sequential decision problem from the financial domain: how to allocate stock orders across a variety of marketplaces, each with an unknown volume available, so as to maximize the number of orders that are filled.

the fact that the number of distinct states in this problem is enormous: it is exponential in the number of venues. They exploit the favorable structure of the problem, and in particular the way it decomposes neatly across the distinct venues. Their approach involves a modification of a conventional nonparametric statistical estimator for censored data—the Kaplan-Meier estimator, which is used in survival analysis. They use one of these estimators for each venue in order to decide on the allocation. Their modification to the Kaplan-Meier estimator incorporates the optimism heuristic by encouraging exploration near the boundary of the region of state space that has already been adequately explored. The key result is that this approach can successfully adapt to unknown markets: under the assumption that the volumes available in the various venues are independent random variables, they prove that their strategy rapidly performs almost as well as the optimal allocation.

The heuristic of optimism in the face of uncertainty is known to perform well in other sequential decision problems. For instance, in small Markov decision problems, it leads to learning algorithms that have small regret: the amount of utility gathered per time step rapidly approaches the best possible value. The key challenge in this area is the development of methods that can deal with very large problems: in a wide range of applications, the state space is enormous; the dark pools problem is typical in this regard. For good performance in a case like this, it seems essential that the problem exhibits some kind of helpful structure. The work detailed in the following paper shows how the generic approach of optimism in the face of uncertainty can be applied to exploit the structure of a very large sequential decision problem to give an adaptive strategy that allows automatic performance optimization. This is an approach that will certainly see wider application. 

Peter L. Bartlett is a professor in the Computer Science Division and Department of Statistics at the University of California, Berkeley.

© 2010 ACM 0001-0782/10/0500 \$10.00

Censored Exploration and the Dark Pool Problem

By Kuzman Ganchev, Yuriy Nevmyvaka, Michael Kearns, and Jennifer Wortman Vaughan

Abstract

Dark pools are a recent type of stock exchange in which information about outstanding orders is deliberately hidden in order to minimize the market impact of large-volume trades. The success and proliferation of dark pools have created challenging and interesting problems in algorithmic trading—in particular, the problem of optimizing the allocation of a large trade over multiple competing dark pools. In this work, we formalize this optimization as a problem of multi-venue exploration from censored data, and provide a provably efficient and near-optimal algorithm for its solution. Our algorithm and its analysis have much in common with well-studied algorithms for managing the exploration–exploitation trade-off in reinforcement learning. We also provide an extensive experimental evaluation of our algorithm using dark pool execution data from a large brokerage.

1. INTRODUCTION

Dark pools are a relatively new type of exchange designed to address the problems that arise from the transparent (or “light”) nature of a typical stock exchange—namely, the difficulty of minimizing the impact of large-volume trades.^{3,5,7} In a typical exchange, the revelation that there is a large-volume buyer (seller) in the market can cause prices to rise (fall) at the buyer’s (seller’s) expense. If the volume is sufficiently large, and the trading period sufficiently short, such market impacts remain even if one attempts to fragment the trade over time into smaller transactions. As a result, there has been increasing interest in recent years in execution mechanisms that allow full or partial concealment of large trades.

In a typical dark pool, buyers and sellers submit orders that simply specify the total volume of shares they wish to buy or sell, with the price of the transaction determined exogenously by “the market”.^a Upon submitting an order to buy (or sell) v shares, a trader is put in a queue of buyers (or sellers) awaiting transaction. Matching between buyers and sellers occurs in sequential arrival of orders, similar to a light exchange. However, unlike a light exchange, no information is provided to traders about how many parties or shares might be available in the pool at any given moment. Thus in a given time period, a submission of v shares results only in a report of how many shares up to v were executed.

While presenting their own trading challenges, dark pools have become tremendously popular exchanges, responsible

for executing 10–20% of the overall US equity volume. In fact, they have been so successful that there are now approximately 40+ dark pools for the US Equity market alone. The popularity of these exchanges has left large-volume traders and brokerages facing a novel problem: How should one optimally distribute a large trade over the many independent dark pools?

To answer this question, we analyze a framework and algorithm for a more general multi-venue exploration problem. We consider a setting in which at each time period, we have some exogenously determined *volume* of V units of an abstract good (for example, shares of a stock that a client would like to sell). Our goal is to “sell” or “consume” as many of these units as possible at each step, and there are K abstract “venues” (for example, various dark pools) in which this selling or consumption may occur. We can divide our V units into any way we like across the venues in service of this goal. What differentiates this problem from most standard learning settings is that if v_i units are allocated to venue i , and all of them are consumed, we learn only that the total demand at venue i was *at least* v_i , not the precise number of units that *could* have been consumed there. This important aspect of our framework is known as *censoring* in the statistics literature.

In this work, we make the natural and common assumption that the maximum amount of consumption available in venue i at each time step (or the total liquidity available, in the dark pool problem) is drawn according to a fixed but unknown distribution P_i . Formally speaking, this means that when v_i units are submitted to venue i , a value s_i is drawn randomly from P_i and the observed (and possibly censored) amount of consumption is $\min\{s_i, v_i\}$.

A learning algorithm in our framework receives a sequence of volumes $V^1, V^2, \dots$ and must decide how to distribute the V^t units across the venues at each time step t . Our goal is to *efficiently* (in time polynomial in the parameters of the model) learn a near-optimal allocation policy. There is a distinct *between-venue exploration* component to this problem, since the best number of shares to submit to venue i may depend on both V^t and the distributions for the *other* venues, and the only mechanism by which we can discover the distributions is by submitting allocations. If we routinely submit too-small volumes to a venue, we receive censored observations and are underutilizing the venue; if we submit

^a For our purposes, we can think of the price as the midpoint between the bids and ask in the light exchanges, though this is a slight oversimplification.

The original version of this paper was published in the *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009.

too-large volumes, we receive uncensored observations but have excess inventory.

Our main theoretical contribution is a provably polynomial-time algorithm for learning a near-optimal policy for any unknown venue distributions P_i . This algorithm takes a particularly natural and appealing form, in which *allocation* and *distribution reestimation* are repeatedly alternated. More precisely, at each time step we maintain estimates of the distributions P_i ; pretending that these estimates are in fact exactly correct, we allocate the current volume V accordingly. These allocations generate observed consumptions in each venue, which in turn are used to update the estimates. We show that when the estimates are “optimistic tail modifications” of the classical *Kaplan–Meier* maximum likelihood estimator for censored data, this estimate–allocate loop has *provably efficient between-venue exploration* behavior that yields the desired result. Venues with smaller available volumes are gradually given smaller allocations in the estimate–allocate loop, whereas venues with repeated censored observations are gradually given larger allocations, eventually settling on a near-optimal overall allocation distribution.

Finally, we present an extensive experimental evaluation of our model and algorithm on the dark pool problem, using trading data from a large brokerage.

The closest problem to our setting is the widely studied *newsvendor problem* from the operations research literature. In this problem, at each time period a player (representing a newsstand owner) chooses a quantity V of newspapers to purchase at a fixed per-unit price, and tries to optimize profit in the face of demand uncertainty at a single venue (their newsstand).^b Huh et al.¹⁰ were the first to consider the use of the Kaplan–Meier estimator in this class of problems. They use an estimate–allocate loop similar to ours, and show *asymptotic* convergence to near-optimal behavior in a single venue. Managing the distribution of an *exogenously specified* volume V across *multiple* venues (which are the important aspects of the dark pool problem, where the volume to be traded is specified by a client, and there are many dark pools) and the attendant exploration–exploitation trade-off *between venues* are key aspects and differentiators of our algorithm and analysis. We also obtain stronger (polynomial time rather than asymptotic) bounds, which require a modification of the classical Kaplan–Meier estimator.

2. THE FORMAL MODEL

Formally, we consider the following problem. At each time t , a learner is presented with a quantity or *volume* $V^t \in \{1, \dots, V\}$ of units, where V^t is sampled from an unknown distribution Q . The learner must decide on an *allocation* $\vec{v}^t$ of these shares to a set of K known *venues*, with $v_i^t \in \{0, \dots, V^t\}$ for each $i \in \{1, \dots, K\}$, and $\sum_{i=1}^K v_i^t = V^t$. The learner is then told the number of units r_i^t consumed at each venue i . Here $r_i^t = \min\{s_i^t, v_i^t\}$, where s_i^t is the maximum consumption level of venue i at time t , which is sampled independently

^b In our setting, it is important that we view V as given exogenously by the client and not under the trader’s control, which distinguishes our setting somewhat from the prior works.

from a fixed but unknown distribution P_i . If $r_i^t = v_i^t$, we say that the algorithm receives a *censored observation* because it is possible to infer only that $r_i^t \leq s_i^t$. If $r_i^t < v_i^t$, we say that the algorithm receives a *direct observation* because it must be the case that $r_i^t = s_i^t$.

The goal of the learner is to discover a near-optimal one-step allocation policy, that is, an allocation policy that approximately optimizes the expected number of units out of V^t consumed at each time step t . (We briefly discuss other objectives at the end of Section 4.4.)

Throughout the remainder of the paper, we use the shorthand T_i for the *tail probabilities* associated with P_i . That is, $T_i(s) = \sum_{s' \geq s} P_i(s')$.^c Clearly $T_i(0) = 1$ for all i . We use $\hat{T}_i^t(s)$ for an empirical estimate of $T_i(s)$ at time t .

3. A GREEDY ALLOCATION SCHEME

Before tackling the full exploration–exploitation problem, we must examine a more basic question: Given estimates $\hat{T}_i$ of the tail probabilities T_i for each venue i , how can we maximize the (estimated) expected number of units consumed on a single time step? It turns out that this can be accomplished using a simple greedy allocation scheme. The greedy algorithm allocates one unit at a time. The venue to which the next unit is allocated is chosen to maximize the estimated probability that the unit will be consumed. It is easy to see that if v_i units have already been allocated to venue i , then the estimated probability that the next allocated unit will be consumed is simply $\hat{T}_i(v_i + 1)$. A formal description of the Greedy algorithm is given in Figure 1.

THEOREM 1. *The allocation returned by Greedy maximizes the expected number of units consumed in a single time step, where the expectation is taken with respect to the estimated tail probabilities $\{\hat{T}_i\}_{i=1}^K$.*

The proof of this theorem is fairly simple. Using the fact that tail probabilities must satisfy $\hat{T}_i(s) \geq \hat{T}_i(s')$ for all $s \leq s'$, it is easy to verify that by greedily adding units to the venues in decreasing order of $\hat{T}_i(s)$, the algorithm returns

$$\arg \max_{\vec{v}} \sum_{i=1}^K \sum_{s=1}^{v_i} \hat{T}_i(s) \quad \text{s.t.} \quad \sum_{i=1}^K v_i = V.$$

The remainder of the proof involves showing that the expression being maximized here equivalent to the expected number of units consumed. This can be done algebraically.^d

4. THE CENSORED EXPLORATION–EXPLOITATION ALGORITHM

We now present our main theoretical result, which is a

^c In the early literature on censored estimation, these tail probabilities were referred to as *survival probabilities*, as $T(s)$ usually represented the probability that a patient in a particular medical study survived for at least s years past the start of the study. In this setting, observations were frequently censored when researchers lost track of a patient midway through the study and knew only that the patient lived *at least* until the point at which contact was broken.¹

^d The curious reader can find more details of this and other omitted proofs in the original version of this paper.⁹

Figure 1. Optimal allocation algorithm *Greedy*.

```

Input: Volume  $V$ , tail probability estimates  $\{\hat{T}_i\}_{i=1}^K$ 
Output: An allocation  $\bar{v}$ 
 $\bar{v} \leftarrow \vec{0}$ ;
for  $\ell \leftarrow 1$  to  $V$  do
   $i \leftarrow \operatorname{argmax}_i \hat{T}_i(v_i + 1)$ ;
   $v_i \leftarrow v_i + 1$ ;
end
return  $\bar{v}$ 

```

polynomial-time, near-optimal algorithm for multi-venue exploration from censored data. The analysis of our algorithm bears strong resemblance to the exploration-exploitation arguments common in the E^3 and RMAX family of algorithms for reinforcement learning.^{4, 12} In particular, there is an analogy to the notion of a *known state* inherent in those earlier algorithms, along with an *exploitation lemma* (proving that expected payoffs from known states are high) and an *exploration lemma* (proving that extended periods of low payoffs must result in more states becoming known). In our setting, however, the number of states is exponential and thus the special structure of our problem is required to obtain a polynomial time algorithm. We first provide an overview of the algorithm and its analysis before examining it in more detail.

At the highest level, the algorithm is quite simple and natural. It maintains estimates $\hat{T}_i^t$ for the true unknown tail probabilities T_i for each venue i . These estimates improve with time in a particular quantifiable sense which drives between-venue exploration. At any given time t , the current volume V^t is allocated across the venues by simply calling the optimal greedy allocation scheme from Figure 1 on the current set of estimated tail probabilities $\hat{T}_i^t$. This results in new censored observations from each venue, which in turn are used to update the estimates $\hat{T}_i^{t+1}$ used at the next time step. Thus the algorithm, which is formally stated in Figure 2, implements a continuous allocate-reestimate loop.

Note that we have not yet described the algorithm's subroutine *OptimisticKM*, which specifies how we estimate $\hat{T}_i^t$ from the observed data. The most natural choice would be the maximum likelihood estimator on the data. This estimator is well-known in the statistics literature as the *Kaplan-Meier* estimator. In the following section, we describe Kaplan-Meier and derive a new convergence result that suits our particular needs. This result in turn lets us define an optimistic tail modification of Kaplan-Meier that becomes our choice for *OptimisticKM*. Figure 3 shows the full subroutine.

The analysis of our algorithm, which is developed in more detail over the next few sections, proceeds as follows:

Step 1: We first review the Kaplan-Meier maximum likelihood estimator for censored data and provide a new finite sample convergence bound for this estimator. This bound allows us to define a *cut-off* for each venue i such that the Kaplan-Meier estimate of the tail probability $T_i(s)$

for every value of s up to the cut-off is guaranteed to be close to the true tail probability. We then define a lightly modified version of the Kaplan-Meier estimates in which the tail probability of the next unit above the cut-off is modified in an optimistic manner. We show that in conjunction with the greedy allocation scheme, this minor modification leads to increased exploration, since the next unit beyond the cut-off always looks at least as good as the cut-off itself.

Step 2: We next prove our main *Exploitation Lemma* (Lemma 3). This lemma shows that at any time step, if it is the case that the number of units allocated to each venue by the greedy algorithm is strictly below the cut-off for that venue (which can be thought of as being in a *known state* in the parlance of reinforcement learning) then the allocation is provably ε -optimal.

Step 3: We then prove our main *Exploration Lemma* (Lemma 4), which shows that on any time step at which the allocation made by the greedy algorithm is *not* ε -optimal, it is possible to lower bound the probability that the algorithm explores. Thus, any time we cannot ensure a near-optimal allocation, we are instead assured of exploring.

Step 4: Finally, we show that on any sufficiently long sequence of time steps (where *sufficiently long* is polynomial in the parameters of the model), it must be the case that either the algorithm has already implemented a near-

Figure 2. Main algorithm.

```

Input: Volume sequence  $V^1, V^2, V^3, \dots$ 
Arbitrarily initialize  $\hat{T}_i^1$  for each  $i$ ;
for  $t \leftarrow 1, 2, 3, \dots$  do
  % Allocation Step:
   $\bar{v}^t \leftarrow \text{Greedy}(V^t, \hat{T}_1^t, \dots, \hat{T}_K^t)$ ;
  for  $i \leftarrow \{1, \dots, K\}$  do
    Submit  $V_i^t$  units to venue  $i$ ;
    Let  $r_i^t$  be the number of shares sold;
  % Reestimation Step:
   $\hat{T}_i^{t+1} \leftarrow \text{OptimisticKM}(\{(V_i^t, r_i^t)\}_{t=1}^t)$ ;
end
end

```

Figure 3. Subroutine *OptimisticKM*. Let $M_{i,s}^t$ and $N_{i,s}^t$ be defined in Section 4.1, and assume that $\varepsilon, \delta > 0$ are fixed parameters.

```

Input: Observed data  $\{(V_i^t, r_i^t)\}_{t=1}^t$  for venue  $i$ 
Output: Modified Kaplan-Meier estimators for  $i$ 
% Calculate the cut-off:
 $c_i^t \leftarrow \max\{s : s = 0 \text{ or } N_{i,s-1}^t \geq 128 (sV/\varepsilon)^2 \ln(2V/\delta)\}$ ;
% Compute Kaplan-Meier tail probabilities:
 $\hat{T}_i^t(0) = 1$ ;
for  $s = 1$  to  $V$  do
   $\hat{T}_i^t(s) \leftarrow \prod_{s'=0}^{s-1} (1 - (M_{i,s'}^t / N_{i,s'}^t))$ ;
end
% Make the optimistic modification:
if  $c_i^t < V$  then
   $\hat{T}_i^t(c_i^t + 1) \leftarrow \hat{T}_i^t(c_i^t)$ ;
return  $\hat{T}_i^t$ ;

```

optimal solution at almost every time step (and thus will continue to perform well in the future), or the algorithm has explored sufficiently often to learn accurate estimates of the tail distributions out to V units on every venue. In either case, we can show that with high probability, at the end of the sequence, the current algorithm achieves an ε -optimal solution at each time step with probability at least $1 - \varepsilon$.

4.1. Convergence of Kaplan–Meier estimators

We begin by describing the standard Kaplan–Meier maximum likelihood estimator for censored data,^{11, 13} restricting our attention to a single venue i . Let $z_{i,s}$ be the true probability that the demand in this venue is *exactly* s units given that the demand is *at least* s units. Formally,

$$z_{i,s} = \frac{T_i(s) - T_i(s+1)}{T_i(s)} = 1 - \frac{T_i(s+1)}{T_i(s)}.$$

It is easy to verify that for any $s > 0$,

$$T_i(s) = \prod_{s'=0}^{s-1} \frac{T_i(s'+1)}{T_i(s')} = \prod_{s'=0}^{s-1} (1 - z_{i,s'}).$$

At a high level, we can think of Kaplan–Meier as first computing a separate estimate of $z_{i,s}$ for each s and then using these estimates to compute an estimate of $T_i(s)$.

More specifically, let $M_{i,s}^t$ be the number of *direct* observations of s units up to time t , that is, the number of time steps at which strictly more than s units were allocated to venue i and exactly s were consumed. Let $N_{i,s}^t$ be the number of either direct or censored observations of *at least* s units on time steps at which strictly more than s units were allocated to venue i . We can then naturally define our estimate $\hat{z}_{i,s}^t = M_{i,s}^t / N_{i,s}^t$, with $\hat{z}_{i,s}^t = 0$ if $N_{i,s}^t = 0$. The Kaplan–Meier estimator of the tail probability for any $s > 0$ after t time steps can then be expressed as

$$\hat{T}_i^t(s) = \prod_{s'=0}^{s-1} (1 - \hat{z}_{i,s'}^t), \quad (1)$$

with $\hat{T}_i^t(0) = T_i(0) = 1$ for all t .

Previous work has established convergence rates for the Kaplan–Meier estimator to the true underlying distribution in the case that each submission in the sequence $v_1^t, \dots, v_i^t$ is independently and identically distributed (i.i.d.),⁸ and asymptotic convergence for non-i.i.d. settings.¹⁰ We are not in the i.i.d. case, since the submitted volumes at one venue are a function of the entire history of allocations and executions across all venues. In the following theorem, we give a new finite sample convergence bound applicable to our setting.

THEOREM 2. *Let $\hat{T}_i^t$ be the Kaplan–Meier estimate of T_i as given in Equation 1. For any $\delta > 0$, with probability at least $1 - \delta$, for every $s \in \{1, \dots, V\}$,*

$$|T_i(s) - \hat{T}_i^t(s)| \leq s \sqrt{2 \ln(2V/\delta) / N_{i,s-1}^t}.$$

This result shows that as we make more and more direct or censored observations of at least $s - 1$ units on time steps at which at least s units are allocated to venue i , our estimate

of the tail probability for s shares rapidly improves.

To prove this theorem, we must first show that the estimates $\hat{z}_{i,s}^t$ converge to the true probabilities $z_{i,s}$. In an i.i.d. setting, this could be accomplished easily using standard concentration results such as Hoeffding’s inequality. In our setting, we instead appeal to Azuma’s inequality (see, for example, Alon and Spencer²), a tool for bounding *martingales*, or sequences $X_1, X_2, \dots$ such that for each n , $|X_n - X_{n+1}| \leq 1$ and $E[X_{n+1} | X_n] = X_n$. In particular, we show that the value $N_{i,s}^t (z_{i,s} - \hat{z}_{i,s}^t)$ can be expressed as the final term of a martingale sequence, allowing us to bound its absolute value. This in turn implies that bound on $|z_{i,s} - \hat{z}_{i,s}^t|$ that we need, and all that remains is to show that these bounds imply a bound on the discrepancy between $T_i(s)$ and the estimator $\hat{T}_i^t(s)$.

4.2. Modifying Kaplan–Meier

In Figure 3, we describe the minor modification of Kaplan–Meier necessary for our analysis. As described above (Step 1), the value c_i^t in this algorithm can intuitively be viewed as a cut-off up to which we are guaranteed to have sufficient data to accurately estimate the tail probabilities using Kaplan–Meier; this is formalized in Lemma 1. Thus for every quantity $s < c_i^t$, we simply let $\hat{T}_i^t(s)$ be precisely the Kaplan–Meier estimate as in Equation 1.

However, to promote exploration, we set the value of $\hat{T}_i^t(c_i^t + 1)$ optimistically to the Kaplan–Meier estimate of the tail probability at c_i^t (not at $c_i^t + 1$). This optimistic modification is necessary to ensure that the greedy algorithm explores (i.e., has a chance of making progress towards increasing at least one cut-off value) on every time step for which it is not already producing an ε -optimal allocation. In particular, suppose that the current greedy solution allocated no more than c_i^t units to any venue i and exactly c_j^t units to some venue j . Using the standard Kaplan–Meier tail probability estimates, it could be the case that this allocation is suboptimal (there is no way to know if it would have been better to include unit $c_i^t + 1$ from venue j in place of a unit from another venue since we do not have an accurate estimate of the tail probability for this unit), and yet no exploration is taking place. By optimistically modifying the tail probability $\hat{T}_i^t(c_i^t + 1)$ for each venue, we ensure that no venue remains unexplored simply because the algorithm unluckily observes a low demand a small number of times.

We now formalize the idea of c_i^t as a cut-off up to which the Kaplan–Meier estimates are accurate. In the results that follow, we think of $\varepsilon > 0$ and $\delta > 0$ as fixed parameters of the algorithm.^c

LEMMA 1. *For any $s \leq V$, let $\hat{T}_i^t(s)$ be the Kaplan–Meier estimator for $T_i(s)$ returned by OptimisticKM. With probability at least $1 - \delta$, for all $s \leq c_i^t$, $|T_i(s) - \hat{T}_i^t(s)| \leq \varepsilon / (8V)$.*

PROOF. It is always the case that $T_i(0) = \hat{T}_i^t(0) = 1$, so the result

^c In particular, ε corresponds to the value ε specified in Theorem 3, and δ corresponds roughly to that δ divided by the polynomial upper bound on time steps.

holds trivially unless $c_i^t > 0$. Suppose this is the case. Recall that $N_{i,s}^t$ is the number of direct or censored observations of at least s units on time steps at which strictly more than s units were allocated to venue i . By definition, it must be the case that $N_{i,s}^t \geq N_{i,s'}^t$, whenever $s \leq s'$. Thus by definition of the cut-off c_i^t in Figure 3, for all $s < c_i^t$, $N_{i,s}^t \geq 128(sV/\varepsilon)^2 \ln(2V/\varepsilon)$. The lemma then follows immediately from an application of Theorem 2. $\square$

Lemma 2 shows that it is also possible to achieve additive bounds on the error of tail probability estimates for quantities s much *bigger* than c_i^t as long as the estimated tail probability at c_i^t is sufficiently small. Intuitively, this is because the tail probability at these large values of s must be smaller than the true tail probability at c_i^t , which, in this case, is known to be very small already.

LEMMA 2. *If $\hat{T}_i^t(c_i^t) \leq \varepsilon/(4V)$ and the high probability event in Lemma 1 holds, then for all s such that $c_i^t < s \leq V$, $|T_i(s) - \hat{T}_i^t(s)| \leq \varepsilon/(2V)$.*

4.3. Exploitation and exploration lemmas

We are now ready to state our main *Exploitation Lemma* (Step 2), which formalizes the idea that once a sufficient amount of exploration has occurred, the allocation output by the greedy algorithm is ε -optimal. The proof of this lemma is where the optimistic tail modification to the Kaplan–Meier estimator becomes important. In particular, because of the optimistic setting of $\hat{T}_i^t(c_i^t + 1)$, we know that if the greedy policy allocates exactly c_i^t units to a venue i , it could not gain too much by reallocating additional units from another venue to venue i instead. In this sense, we create a buffer above each cut-off, guaranteeing that it is not necessary to continue exploring as long as one of the two conditions in the lemma statement is met for each venue.

The second condition in the lemma may appear mysterious at first. To see why it is necessary, notice that the rate at which the estimate $\hat{T}_i^t(c_i^t + 1)$ converges to the true tail probability $T_i(c_i^t + 1)$ implied by Theorem 2 depends on the number of times that we observe a consumption of c_i^t or more units. If $T_i(c_i^t)$ is very small, then the consumption of this many units does not frequently occur. Luckily, if this is the case, then we know that $T_i(c_i^t + 1)$ must be very small as well, and more exploration of this venue is not needed.

LEMMA 3 (EXPLOITATION LEMMA). *Assume that at time t , the high probability event in Lemma 1 holds. If for each venue i , either (1), $v_i^t \leq c_i^t$ or (2), $\hat{T}_i^t(c_i^t) \leq \varepsilon/(4V)$, the difference between the expected number of units consumed under allocation $\vec{v}^t$ and the expected number of units consumed under the optimal allocation is at most ε .*

PROOF SKETCH. The proof begins by creating an arbitrary one-to-one mapping between the units allocated to different venues by the algorithm and an optimal allocation. Consider any such pair in this mapping.

If the first condition in the lemma holds for the venue i to which the unit was allocated by the algorithm, we can use Lemma 1 to show that the algorithm's estimate of the probability of this unit being consumed is close to the true

probability; in particular, the algorithm is not overestimating this probability too much. If the second condition holds, then the algorithm's estimate of the probability of the share being consumed is so small that, again, the algorithm cannot possibly be overestimating it too much (because the lowest the probability could be is zero). This follows from Lemma 2.

Now consider the venue j to which unit was allocated by the optimal allocation. If the number of units v_j^t allocated to this venue by the algorithm is strictly less than the cut-off c_j^t , then by Lemma 1, the algorithm could not have underestimated the probability of additional units being consumed by too much. Furthermore, because of the optimistic tail modification of the Kaplan–Meier estimator, this also holds if $v_j^t = c_j^t$. Finally, if it is instead the case that the second condition in the lemma statement holds for venue j , then the algorithm again could not possibly have underestimated the probability of the unit being consumed too much because the true probability is so low.

Putting these pieces together, we can argue that for each pair in the matching (of which there are no more than V), since the algorithm did not overestimate the probability of unit it chose being consumed by too much (in this case, *too much* means more than $\varepsilon/(2V)$) and did not underestimate the probability of the corresponding unit in the optimal allocation by too much (again, by $\varepsilon/(2V)$), the difference in expected units consumed between the optimal allocation and the algorithm's is at most ε . $\square$

Finally, Lemma 4 presents the main exploration lemma (Step 3), which states that on any time step at which the allocation is *not* ε -optimal, the probability of obtaining a useful observation is at least $\varepsilon/(8V)$.

LEMMA 4 (EXPLORATION LEMMA). *Assume that at time t , the high probability event in Lemma 1 holds. If the allocation is not ε -optimal, then for some venue i , with probability at least $\varepsilon/(8V)$, $N_{i,c_i^t}^{t+1} = N_{i,c_i^t}^t + 1$.*

PROOF. Suppose the allocation is not ε -optimal at time t . By Lemma 3, it must be the case that there exists some venue i for which $v_i^t > c_i^t$ and $\hat{T}_i^t(c_i^t) > \varepsilon/(4V)$, i.e., a venue in which the algorithm has allocated units past the cut-off but for which the tail probability at the cut-off is not too close to zero. Let i be a venue for which this is true. Since $v_i^t > c_i^t$, it will be the case that the algorithm obtains a useful observation for exploration of this venue (i.e., an observation causing $N_{i,c_i^t}^t$ to be incremented) if the number of units consumed at this venue is sufficiently high (specifically, if $r_i^t > c_i^t$). Since $\hat{T}_i^t(c_i^t) > \varepsilon/(4V)$, Lemma 1 implies that $T_i(c_i^t) > \varepsilon/(8V)$, which in turn implies that the number of units consumed is high enough to constitute a useful observation with probability at least $\varepsilon/(8V)$. $\square$

4.4. Putting it all together

With the exploitation and exploration lemmas in place, we are finally ready to state our main theorem.

THEOREM 3 (MAIN THEOREM). *For any $\varepsilon > 0$ and $\delta > 0$, with probability $1 - \delta$ (over the randomness of draws from Q and $\{P_i\}$), after running for a time polynomial in K , V , $1/\varepsilon$, and $\ln(1/\delta)$, the algorithm in Figure 2 makes an ε -optimal allocation*

on each subsequent time step with probability at least $1 - \varepsilon$.

PROOF SKETCH. Suppose that the algorithm runs for R time steps, where R is a (specific, but unspecified for now) polynomial in the model parameters K , V , $1/\varepsilon$, and $\ln(1/\delta)$. If it is the case that the algorithm was already ε -optimal on a fraction $(1 - \varepsilon)$ of the R time steps, then we can argue that the algorithm will continue to be ε -optimal on at least a fraction $(1 - \varepsilon)$ of future time steps since the algorithm's performance should improve on average over time as estimates become more accurate.

On the other hand, if the algorithm chose sub-optimal allocations on at least a fraction ε of the R time steps, then by Lemma 4, the algorithm must have incremented $N_{i,c_i^t}^t$ for some venue i and cut-off c_i^t approximately $\varepsilon^2 R/(8V)$ times. By definition of the c_i^t , it can never be the case that $N_{i,c_i^t}^t$ was incremented too many times for any *fixed* values of i and c_i^t (where *too many* is a polynomial in V , $1/\varepsilon$, and $\ln(1/\delta)$); otherwise the cut-off would have increased. Since there are only K venues and V possible cut-off values to consider in each venue, the total number of increments can be no more than KV times this polynomial, another polynomial in V , $1/\varepsilon$, $\ln(1/\delta)$, and now K . If R is sufficiently large (but still polynomial in all of the desired quantities) and approximately $\varepsilon^2 R/(8V)$ increments were made, we can argue that *every* venue must have been fully explored, in which case, again, future allocations will be ε -optimal. $\square$

We remark that our optimistic tail modifications of the Kaplan–Meier estimators are relatively mild. This leads us to believe that using the same estimate–allocate loop with an *unmodified* Kaplan–Meier estimator would frequently work well in practice. We investigate a parametric version of this learning algorithm in the experiments described below.

5. THE DARK POOL PROBLEM

The remainder of this article is devoted to the application of our techniques to the dark pool problem. We begin with a description of the trading data we used, and go on to describe a variety of experiments we performed.

5.1. Summary of the dark pool data

Our data set is from the internal dark pool order flow for a major US broker–dealer. Each (possibly censored) observation is of the form discussed throughout the paper—a triple consisting of the dark pool name, the number of shares sent to that pool, and the number of shares subsequently executed within a short time interval. It is important to highlight some limitations of the data. First, note that the data set conflates the policy the brokerage used for allocation across the dark pools with the liquidity available in the pools themselves. For our data set, the policy in force was very similar to the bandit-style approach we discuss below. Second, the “parent” orders determining the overall volumes to be allocated across the pools were determined by the brokerage's trading needs, and are similarly out of our control.

The data set contains submissions and executions for four active dark pools: BIDS Trading, Automated Trading Desk, D.E. Shaw, and NYFIX, each for a dozen of relatively actively-traded stocks,^f thus yielding 48 distinct

stock–pool data sets. The average daily trading volume of these stocks across all exchanges (light and dark) ranges from 1 to 60 million shares, with a median volume of 15 million shares. Energy, Financials, Consumer, Industrials, and Utilities industries are represented. Our data set spans 30 trading days. For every stock–pool pair we have on average 1,200 orders (from 600 to 2,000), which corresponds to 1.3 million shares (from 0.5 to 3 million). Individual order sizes range from 100 to 50,000 shares, with 1,000 shares being the median. Sixteen percent of orders are filled at least partially (meaning that fully 84% result in no shares executed), 9% of the total submitted volume was executed, and 11% of all observations were censored.

5.2. Parametric models for dark pools

The theory and algorithm we have developed for censored exploration permit a very general form for the venue distributions P_i . The downside of this generality is that we are left with the problem of learning a very large number of parameters. More parameters generally mean that more data is necessary to guarantee that the model will generalize well, which means more rounds of exploration are needed before the algorithm's future performance is near-optimal. In some applications, it is therefore advantageous to employ a less general but more simple parametric form for these distributions.

We experimented with a variety of common parametric forms for the distributions. For each such form, the basic methodology was the same. For each of the $4 \times 12 = 48$ venue–stock pairs, the data for that pair was split evenly into a training set and a test set. The training data was used to select the maximum likelihood model from the parametric class. Note that we can no longer directly apply the nonparametric Kaplan–Meier estimator—within each model class, we must directly maximize the likelihood on the censored training data. This is a relatively straightforward and efficient computation for each of the model classes we investigated. The test set was then used to measure the generalization performance of each maximum likelihood model.

Our investigations revealed that the best models maintained a separate parameter for the probability of zero shares being available (that is, $P_i(0)$ is explicitly estimated)—a *zero bin* or ZB parameter. This is due to the fact that the vast majority of submissions (84%) to dark pools result in no shares being executed. We then examined various parametric forms for the nonzero portions of the venue distributions, including uniform (which of course requires no additional parameters), and Poisson, exponential and power law forms (each of which requires a single additional parameter); each of these forms were applied up to the largest volume submitted in the data sets, then normalized.

The generalization results strongly favor the power law form, in which the probability of s shares being available is proportional to $1/s^\beta$ for real β —a so-called heavy-tailed

^f Tickers represented are AIG, ALO, CMI, CVX, FRE, HAL, JPM, MER, MIR, NOV, XOM, and NRG.

Table 1. Average per-sample log-loss (negative log likelihood) for each venue distribution models. The “Wins” column shows the number of stock-venue pairs where a given model beats the other four on the test data.

Model	Train Loss	Test Loss	Wins
Nonparametric	0.454	0.872	3
ZB + Uniform	0.499	0.508	12
ZB + Power Law	0.467	0.484	28
ZB + Poisson	0.576	0.661	0
ZB + Exponential	0.883	0.953	5

distribution when $\beta > 0$. Nonparametric models trained with Kaplan–Meier are best on the training data but overfit badly due to their complexity relative to the sparse data, while the other parametric forms cannot accommodate the heavy tails of the data. This is summarized in Table 1. Based on this comparison, for our dark pool study we investigate a variant of our main algorithm, in which the estimate–allocate loop has an estimation step using maximum likelihood estimation within the ZB + Power Law model, and allocations are done greedily on these same models.

In terms of the estimated ZB + Power Law parameters themselves, we note that for all 48 stock–pool pairs the Zero Bin parameter accounted for most of the distribution (between a fraction 0.67 and 0.96), which is not surprising considering the aforementioned preponderance of entirely unfilled orders in the data. The vast majority of the 48 exponents β fell between $\beta = 0.25$ and $\beta = 1.3$ —so rather long tails indeed—but it is noteworthy that for one of the four dark pools, 7 of the 12 estimated exponents were actually *negative*, yielding a model that predicts *higher* probabilities for larger volumes. This is likely an artifact of our size- and time-limited data set, but is not entirely unrealistic and results in some interesting behavior in the simulations.

5.3. Data-based simulation results

As in any control problem, the dark pool data in our possession is unfortunately insufficient to evaluate and compare different allocation algorithms. This is because of the aforementioned fact that the volumes submitted to each venue were fixed by the specific policy that generated the data, and we cannot explore alternative choices—if our algorithm chooses to submit 1000 shares to some venue, but in the data only 500 shares were submitted, we simply cannot infer the outcome of our desired submission.

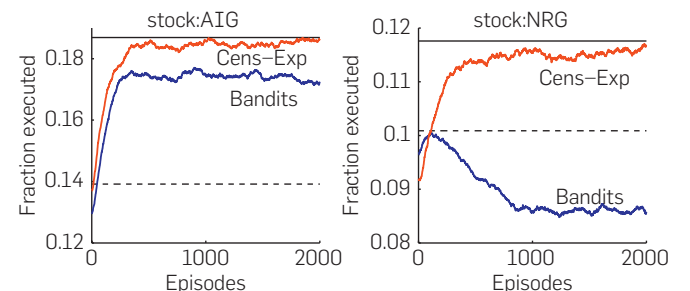
We thus instead use the raw data to derive a *simulator* with which we can evaluate different approaches. In light of the modeling results of Section 5.2, the simulator for stock S was constructed as follows. For each dark pool i , we used *all* of the data for i and stock S to estimate the maximum likelihood Zero Bin + Power Law distribution. (Note that there is no need for a training-test split here, as we have already separately validated the choice of distributional model.) This results in a set of four venue distribution models P_i that form

the simulator for stock S . This simulator accepts allocation vectors (v_1, v_2, v_3, v_4) indicating how many shares some algorithm wishes to submit to each venue, draws a “true liquidity” value s_i from P_i for each i , and returns the vector (r_1, r_2, r_3, r_4) , where $r_i = \min(v_i, s_i)$ is the possibly censored number of shares filled in venue i .

Across all 12 stocks, we compared the performance of four different allocation algorithms. The (obviously unrealistic) *ideal allocation* is given the *true parameters* of the ZB + Power Law distributions used by the simulator and allocates shares optimally (greedily) with respect to these distributions. The *uniform allocation* divides any order equally among all four venues. Our *learning algorithm* implements the repeated allocate–reestimate loop as in Figure 2, using the maximum likelihood ZB + Power Law model for the reestimation step. Finally, the simple (and fairly naive) *bandit-style algorithm* maintains a weighting over the venues and chooses allocations proportional to the weights. It begins with equal weights assigned to all venues, and each allocation to a venue which results in any nonzero number of shares being executed causes that venue’s weight to be multiplied by a constant factor α . (Optimizing α over all stock–pool pairs resulted in a value of $\alpha = 1.05$.)

Some remarks on these algorithms are in order. First, note that the ideal and uniform allocation methods are nonadaptive and are meant to serve as baselines—one of them the best performance we could hope for (ideal), and the other the most naive allocation possible (uniform). Second, note that our algorithm has a distinct advantage in the sense that it is using the correct parametric form, the same being used by the simulator itself. Thus our evaluation of this algorithm is certainly optimistic compared to what should be expected in practice. Finally, note that the bandit algorithm is the crudest type of weight-based allocation scheme of the type that abounds in the no-regret literature⁶; we are effectively forcing our problem into a 0/1 loss setting corresponding to “no shares” and “some shares” being executed. Certainly more sophisticated bandit-style approaches can and should be examined.

Figure 4. Sample learning curves. For the stock AIG (left panel), the naive bandits algorithm (labeled blue curve) beats uniform allocation (dashed horizontal line) but appears to asymptote short of ideal (solid horizontal line). For the stock NRG (right panel), the bandits algorithm actually deteriorates with more episodes, underperforming both the uniform and ideal allocations. For both stocks (and the other 10 in our data set), our algorithm (labeled red curve) performs nearly optimally.



Each algorithm was run in simulation for some number of *episodes*. Each episode consisted of the allocation of a fixed number V of shares—thus the same number of shares is repeatedly allocated by the algorithm, though of course this allocation will change over time for the two adaptive algorithms as they learn. Each episode of simulation results in some fraction of the V shares being executed. Two values of V were investigated—a smaller value $V = 1000$, and the larger and potentially more difficult $V = 8000$.

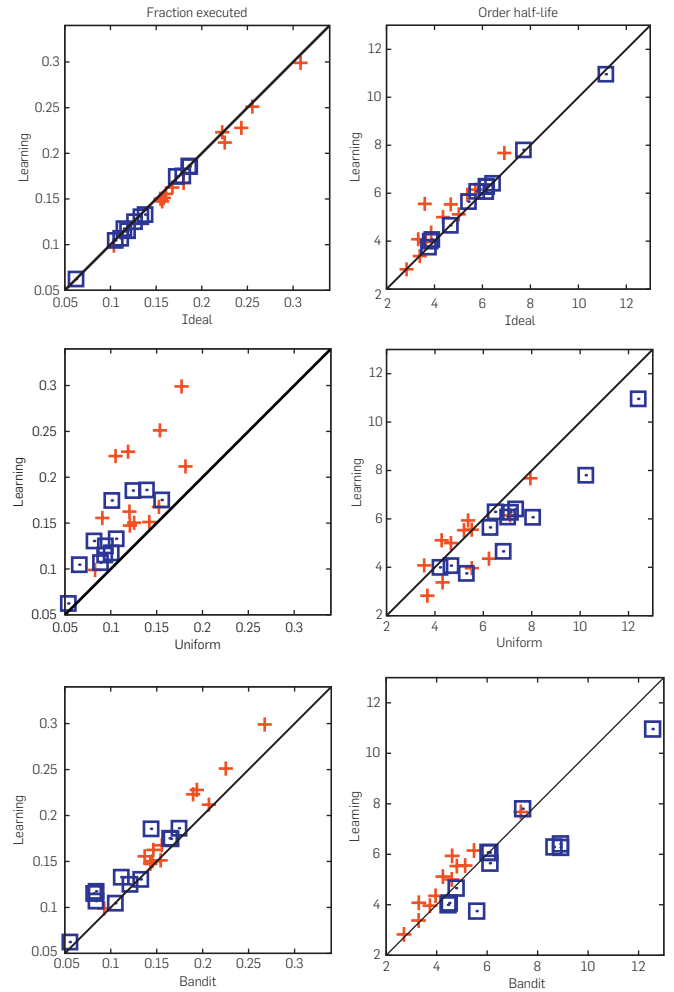
We begin by showing full learning curves over 2000 episodes with $V = 8000$ for a couple of representative stocks in Figure 4. Here the average performance of the two non-adaptive allocation schemes (ideal and uniform) are represented as horizontal lines, while learning curves are given for the adaptive schemes. Due to high variance of the heavy-tailed venue distributions used by the simulator, a single trial of 2000 episodes is extremely noisy, so we both average over 400 trials for each algorithm, and smooth the resulting averaged learning curve with a standard exponential decay temporal moving average.

We see that our learning algorithm converges towards the ideal allocation (as suggested by the theory), often relatively quickly. Furthermore, in each case this ideal asymptote is significantly better than the uniform allocation strawman, meaning that optimal allocations are highly nonuniform. Learning curves for the bandit approach exhibit one of the three general behaviors over the set of 12 stocks. In some cases, the bandit approach is quite competitive with our algorithm, though converging to ideal perhaps slightly slower (not shown in Figure 4). In other cases, the bandit approach learns to outperform uniform allocation but appears to asymptote short of the ideal allocation. Finally, in some cases the bandit approach appears to actually “learn the wrong thing”, with performance decaying significantly with more episodes. This happens when one venue has a very heavy tail, but also a relatively high probability of executing zero shares, and occurs because the very naive bandit approach that we use does not have an explicit representation of the tails of the distribution.

The left column of Figure 5 shows more systematic head-to-head comparisons of our algorithm’s performance versus the other allocation techniques after 2000 episodes for both small and large V . The values plotted are averages of the last 50 points on learning curves similar to Figure 4. These scatterplots show that across all 12 stocks and both settings of V , our algorithm competes well with the optimal allocation, dramatically outperforms uniform, and significantly outperforms the naive bandit allocations (especially with $V = 8000$). The average completion rate across all stocks for the large (small) order sequences is 10.0% (13.1%) for uniform and 13.6% (19.4%) for optimal allocations. Our algorithm performs almost as well as optimal—13.5% (18.7%)—and much better than bandits at 11.9% (17.2%).

In the right column, we measure performance not by the fraction of V shares filled in one step, but by the natural alternative of *order half-life*—the number of steps of

Figure 5. Comparison of our learning algorithm to the three baselines. In each plot, the performance of the learning algorithm is plotted on the y-axis, and the performance of one of the baselines on the x-axis. Left column: Evaluated by the fraction of submitted shares executed in a single time step; higher values are better, and points above the diagonal are wins for our algorithm. Right: Evaluated by order half-life; lower values are better, and points below the diagonal are wins for our algorithm. Each point corresponds to a single stock and order size; small orders (red plus signs) are 1000 shares, large orders (blue squares) are 8000 shares.



repeated resubmission of any remaining shares to get the total number executed above $V/2$. Despite the fact that our algorithm is not designed to optimize this criterion and that our theory does not directly apply to it, we see the same broad story on this metric as well—our algorithm competes with ideal, dominates uniform allocation and beats the bandit approach on large orders. The average order half-life for large (small) orders is 7.2 (5.3) for uniform allocation and 5.9 (4.4) for the greedy algorithm on the true distributions. Our algorithm requires on average 6.0 (4.9) steps, while bandits uses 7.0 (4.4) to trade the large (small) orders.

6. CONCLUSION

While there has been longstanding interest in quantitative finance in the use of models from machine learning and related fields, they are often applied towards the attempt to predict directional price movements, or in the parlance of the field, to “generate alpha” (outperform the market). Here we have instead focused on a problem in what is often called *algorithmic trading*—where one seeks to optimize properties of a specified trade, rather than decide what to trade in the first place—in the recently introduced dark pool mechanism. In part because of the constraints imposed by the mechanism and the structure of the problem, we have been able to adapt and blend methods from statistics and reinforcement learning in the development of a simple, efficient, and provably effective algorithm. We expect there will be many more applications of machine learning methods in algorithmic trading in the future.

Acknowledgments

We are grateful to Curtis Pfeiffer and Andrew Westhead for valuable conversations and to Bobby Kleinberg for introducing us to the literature on the newsvendor problem.



References

1. Akritas, M.G. Nonparametric survival analysis. *Stat. Sci.* 19, 4 (2004), 615–623.
2. Alon, N., Spencer, J. *The Probabilistic Method*, 2nd Edition. Wiley, New York, 2000.
3. Bogoslaw, D. Big traders dive into dark pools. Business Week article, available at: http://www.businessweek.com/investor/content/oct2007/pi2007102_394204.htm, 2007.
4. Brafman, R., Tenenbholz, M. R-MAX—a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.* 3 (2003), 213–231.
5. Carrie, C. Illuminating the new dark influence on trading and U.S. market structure. *J. Trading* 3, 1 (2008), 40–55.
6. Cesa-Bianchi, N., Lugosi, G. *Prediction, Learning, and Games*. Cambridge University Press, 2006.
7. Domowitz, I., Finkelshteyn, I., Yegerman, H. Cul de sacs and highways: an optical tour of dark pool trading performance. *J. Trading* 4, 1 (2009), 16–22.
8. Foldes, A., Rejto, L. Strong uniform consistency for nonparametric survival curve estimators from randomly censored data. *Ann. Stat.* 9, 1 (1981), 122–129.
9. Ganchev, K., Kearns, M., Nevmyvaka, Y., Vaughan, J.W. Censored exploration and the dark pool problem. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009.
10. Huh, W.T., Levi, R., Rusmevichientong, P., Orlin, J. Adaptive data-driven inventory control policies based on Kaplan–Meier estimator. Preprint available at <http://legacy.orie.cornell.edu/~paatrus/psfiles/km-myopic.pdf>, 2009.
11. Kaplan, E.L., Meier, P. Nonparametric estimation from incomplete observations. *J. Am. Stat. Assoc.* 53 (1958), 457–481.
12. Kearns, M., Singh, S. Near-optimal reinforcement learning in polynomial time. *Mach. Learn.* 49 (2002), 209–232.
13. Peterson, A.V. Kaplan–Meier estimator. In *Encyclopedia of Statistical Sciences*. Wiley, 1983.

Kuzman Ganchev (kuzman@cis.upenn.edu), University of Pennsylvania.

Yuriy Nevmyvaka (yuriy@cs.cmu.edu), University of Pennsylvania.

Michael Kearns (mkearns@cis.upenn.edu), University of Pennsylvania.

Jennifer Wortman Vaughan (jenn@seas.harvard.edu), Harvard University.

© 2010 ACM 0001-0782/10/0500 \$10.00



For more information, see
<http://interactions.acm.org>



To apply, please send
a résumé, letter of motivation,
and statement of your vision
for the magazine to:

Dan Olsen,
Search Committee Chair
olsen@cs.byu.edu

ACM's interactions Magazine Seeks Its Next Co-Editors-in-Chief

As ACM's premier magazine on applied computer-human interaction (CHI), *interactions* is designed to keep developers, designers, managers, researchers, and users abreast of the latest tools and ideas emerging from the CHI community—and beyond. This colorful, bi-monthly magazine shares timely articles, stories, and practical content related to the interactions between experiences, people, and technology. Its primary objective is to trace innovative technologies from their R&D beginnings to real-world applications and future potential.

interactions is also the membership magazine for ACM's SIGCHI and as such is distributed to more than 7,000 members worldwide.

The role of co-editor-in-chief is a three-year volunteer position and is assisted by ACM professional staff for the production of the magazine. The position starts in the third quarter of 2010 in preparation for the January–February 2011 issue.

interactions

Technical Perspective

Automated Patching Techniques: The Fix Is In

By Mark Harman

OVER THE PAST 40 years, much effort has been devoted to testing software to find bugs. Testing is difficult because the underlying problem involves undecidable questions such as statement reachability. However, testing cannot be ignored. The National Institute of Standards and Techniques estimated the cost of software failure to the U.S. economy at \$60 billion, indicating that improved software testing could reduce this by at least one-third.⁶

If finding bugs is technically demanding and yet economically vital, how much more difficult yet valuable would it be to automatically fix bugs? This question is answered precisely by the work reported in the following paper by Weimer, Forrest, Le Goues, and Nguyen. The authors use evolutionary computation to evolve patches that fix bugs. Their work is the first to show how Genetic Programming can evolve patches that fix real bugs (an idea first proposed by Arcuri and Yao, who experimented on toy programs³).

There is an increasing realization within the software engineering research and development community that evolutionary computation and related search-based optimization can be used to search for solutions to software problems. Software engineers often face problems typified by enormous spaces of possible requirements, designs, test cases, and code. Search-based optimization allows them to deploy optimization algorithms that automatically search these spaces, guided by fitness functions that encode solution desirability.

The search-based optimization approach has come to be known as Search Based Software Engineering (SBSE).⁴ This approach is widely appealing because it is very generic, applying equally well to many varied software engineering problems. SBSE also comfortably handles the multiple, conflicting, and noisy optimization objectives often found in software

engineering scenarios. Using SBSE, it has proved possible to automate the search for requirements that balance cost and benefit, designs that maximize cohesion and minimize coupling and to find test cases that balance fault finding against execution time.^{1,2,5,7}

Work on SBSE has gathered pace over the past 10 years and there are now over 650 papers^a on various applications of search-based optimization to software engineering problems. The following paper is sure to be looked back upon as a significant and tangible breakthrough in this history of SBSE research. The authors' work demonstrates that an optimization algorithm is capable of repairing broken programs. Since it is automated, it can fix bugs much faster than any human. For example, the Zune bug was patched in a little over three minutes on standard equipment.

Currently, both developers and researchers remain cautious about the idea of deploying evolved program code. However, with the benefit of hindsight, we may look back on this with the same historical perspective that we now apply to compiled code. That is, despite its current widespread use, there was, within living memory, equal skepticism about whether compiled code could be trusted. If a similar change of attitude to evolved code occurs over time, then automated patching will surely be seen to have played a crucial part in this paradigm shift.

The successful application of search-based optimization techniques requires careful configuration and tailoring of the algorithms employed. The more information that can be exploited to guide the search, the better the results. The authors use several innovative techniques to guide their search for patches. Their approach automatically fixes soft-

ware faults in a manner analogous to the way in which a surgeon might go about healing a wound. In order to localize the wound, they use positive and negative test cases, finding code segments most likely to contain the bug. One important insight employed in the patching process is the use of what might be called "plastic surgery;" rather than evolving a patch from scratch, the authors start their evolution from portions of existing code with similar functionality. The evolved patch is subsequently tidied up using techniques for dead code removal, thereby ensuring neat and effective wound healing.

All of these stages are described in detail in the following paper. The authors illustrate the application of their automated patching techniques and give compelling results to show how it finds and fixes real faults. Given the enormous costs of software faults mentioned earlier, it is surely impossible to overstate the potential impact that this work may have. C

References

1. Afzal, W., Torkar, R. and Feldt, R. A systematic review of search-based testing for non-functional system properties. *Info. Softw. Tech.* 51, (2009), 957–976.
2. Ali, S., Briand, L.C., Hemmati, H. and Panesar-Walawege, R.K. A systematic review of the application and empirical investigation of search-based test-case generation. *IEEE Trans. Softw. Eng.* (2010, to appear).
3. Arcuri, A. and Yao, X. A novel co-evolutionary approach to automatic software bug fixing. In *Proceedings of the IEEE Congress on Evolutionary Computation* (Hong Kong, June 1–6, 2008), 162–168.
4. Harman, M. The current state and future of search based software engineering. The Future of Software Engineering. L. Briand and A. Wolf, Eds. *IEEE CS Press*, Alamitos, CA, 342–357.
5. McMinn, P. Search-based software test data generation: A survey. *Softw. Testing, Verification and Reliability* 14, 2 (June 2004), 105–156.
6. National Institute of Standards and Technology. *The Economic Impacts of Inadequate Infrastructure for Software Testing. Planning Report 02–3*, May 2002.
7. Riih , O. A survey on search based software design. Tech. Report Technical Report D-2009-1, Dept. of Computer Sciences, University of Tampere, 2009.

Mark Harman is a professor of software engineering and leads the Software Engineering Group at King's College, London. He is also director of the Centre for Research on Evolution Search and Testing (CREST).

^a Source: SBSE repository at <http://www.sebase.org/sbse/publications/>.

Automatic Program Repair with Evolutionary Computation

By Westley Weimer, Stephanie Forrest, Claire Le Goues, and ThanhVu Nguyen

Abstract

There are many methods for detecting and mitigating software errors but few generic methods for automatically repairing errors once they are discovered. This paper highlights recent work combining program analysis methods with evolutionary computation to automatically repair bugs in off-the-shelf legacy C programs. The method takes as input the buggy C source code, a failed test case that demonstrates the bug, and a small number of other test cases that encode the required functionality of the program. The repair procedure does not rely on formal specifications, making it applicable to a wide range of extant software for which formal specifications rarely exist.

1. INTRODUCTION

Fixing bugs is a difficult, time-consuming, and manual process. Some reports place software maintenance, traditionally defined as any modification made on a system after its delivery, at up to 90% of the total cost of a typical software project.²² Modifying existing code, repairing defects, and otherwise evolving software are major parts of those costs.¹⁹ The number of outstanding software defects typically exceeds the resources available to address them. Mature software projects are forced to ship with both known and unknown bugs¹³ because they lack the development resources to deal with every defect.

In this paper, we describe how to combine this problem by combining program analysis methods with evolutionary computation to automatically repair bugs in off-the-shelf legacy C programs. *Genetic programming* (GP) is a computational method inspired by biological evolution which evolves computer programs tailored to a particular task.¹² GP maintains a population of individual programs, each of which is a candidate solution to the task. Each individual's suitability is evaluated using a task-specific fitness function, and the individuals with highest fitnesses are selected for continued evolution. Computational analogs of biological mutation and crossover produce variations of the high-fitness programs, and the process iterates until a high-fitness program is found. GP has solved an impressive range of problems (e.g., Schmidt and Lipson²¹), but it has not been used to evolve off-the-shelf legacy software. As the 2008 *Field Guide to Genetic Programming* notes, "while it is common to describe GP as evolving programs, GP is not typically used to evolve programs in the familiar Turing-complete languages humans normally use for software development. It is instead more common to evolve programs (or expressions or formulae) in a more constrained and often domain-specific language." (Poli et al.¹⁸ as quoted by Orlov and Sipper¹⁵).

Our approach assumes that we have access to C source code, a negative test case that exercises the fault to be repaired, and

several positive test cases that encode the required behavior of the program. The C program is represented as an abstract syntax tree (AST), in which each node corresponds to an executable statement or control-flow structure in the program. With these inputs in hand, a modified version of GP evolves a candidate repair that avoids failing the negative test case while still passing the positive ones. We then use structural differencing¹ and delta debugging²⁵ techniques to minimize the size of the repair, providing a compact human-readable patch.

A significant impediment for an evolutionary algorithm like GP is the potentially infinite-size search space it must sample to find a correct program. To address this problem we introduce two key innovations. First, we restrict the algorithm so that all variations introduced through mutation and crossover reuse structures in other parts of the program. Essentially, we hypothesize that even if a program is missing important functionality (e.g., a null check) in one location, it likely exhibits the correct behavior in another location, which can be copied and adapted to address the error. Second, we constrain the genetic operations of mutation and crossover to operate only on the region of the program that is relevant to the error, specifically the AST nodes on the execution path that produces the faulty behavior. Instead of searching through the space of all ASTs, the algorithm searches through the much smaller space of nodes representing one execution path. In practice, the faulty execution path has at least an order of magnitude fewer unique nodes than the AST. Combining these insights, we demonstrate automatically generated repairs for eleven C programs totaling 63,000 lines of code.

The main contributions of the work reported in Forrest et al.⁸ and Weimer et al.²⁴ are:

- Algorithms to find and minimize program repairs based on test cases that describe desired functionality. The algorithms are generic in the sense that they can repair many classes of bugs.
- A novel and efficient representation and set of operations for applying GP to program repair. This is the first published work that demonstrates how GP can repair unannotated legacy programs.
- Experimental results showing that the approach gener-

The material in this paper is taken from two original publications, titled "A Genetic Programming Approach to Automated Software Repair" (*Genetic and Evolutionary Computation Conference, 2009*) and "Automatically Finding Patches Using Genetic Programming" (*Proceedings of the 2009 IEEE 31st International Conference on Software Engineering, IEEE Computer Society*).

ates repairs for several classes of defects in 11 production programs taken from multiple domains.

- Experiments to analyze how algorithm performance scales up with problem size and the relative contribution of different components of the evolutionary algorithm.

In the remaining sections of the paper we first give an overview of our technical approach (Section 2), illustrating it with a recent bug in Microsoft's Zune media player (Section 3). In Section 4 we report results obtained for repairs of several benchmark programs and study how algorithm performance scales with problem size. We place the work in the context of prior contributions in Section 5 and discuss our experiences, caveats, and thoughts for future work in Section 6, concluding in Section 7.

2. TECHNICAL APPROACH

The core of our method is an evolutionary algorithm that repairs programs by selectively searching through the space of related program variants until it discovers one that avoids known defects and retains key functionality. We use a novel GP representation and make assumptions about the probable nature and location of the necessary repair, improving search efficiency. Given a defective program, there are several issues to be addressed:

1. **What is it doing wrong?** We take as input a set of *negative test cases* that characterizes a fault. The input program fails all negative test cases.
2. **What is it supposed to do?** We take as input a set of *positive test cases* that encode functionality requirements. The input program passes all positive test cases.
3. **Where should we change it?** We favor changing program locations visited when executing the negative test cases and avoid changing program locations visited when executing the positive test cases.
4. **How should we change it?** We insert, delete, and swap program statements and control flow using existing program structure. We favor insertions based on the existing program structure.
5. **When are we finished?** We call the first variant that passes all positive and negative test cases a primary repair. We minimize the differences between it and the original input program to produce a final repair.

To present the repair process, we first describe our program representation (Section 2.1) and fault localization (Section 2.2) choices. We then detail the GP-based repair strategy (Section 2.3), discussing the genetic operators (Section 2.4), which modify the representation, and the fitness function (Section 2.5), which uses test cases to evaluate the results of the modifications. Finally, a postprocessing step is used to minimize the resulting repair (Section 2.6).

2.1. Representation

There are a number of commonly accepted structures for representing programs, such as control-flow graphs (CFGs) and abstract syntax trees (ASTs). We chose ASTs because they can losslessly represent all structured programs and

tree operations are well studied in GP. ASTs can be expressed at multiple levels of abstraction or granularity, and our program representation reflects the trade-off between expressive power and scalability. For example, C programs contain both *statements*, such as the conditional statement “`if (!p) {x = 0;}`” and *expressions*, such as “`0`” or “`!p`”. For scalability, we treat the statement as the basic unit, or gene. Thus, we never modify “`!p`” into “`(p | error_flag)`” because doing so would involve changing the inner structure of an expression. Instead, when manipulating compound statements, we operate on entire AST subtrees. For example, we might delete the entire “`if ...`” statement, including its then-branch and else-branch children. Finally, we never directly modify low-level control-flow directives such as `break`, `continue`, or `goto`, although statements around them can be modified.

2.2. Fault localization

We assume that software defects are local and that fixing one does not require changing the entire program. This assumption narrows the search space by limiting code changes to portions of the program likely to contain the defect. We bias modifications toward statement nodes that were visited when running the negative test cases but not visited when running the positive test cases. We find this information by assigning each statement a unique ID and instrumenting the program to print out the ID of each statement visited.¹⁴ This allows our approach to scale to larger program sizes. For example, while the `atris` program contains a total of 8068 statement nodes (Table 1), we use this fault localization information to bias the search toward 34 statement nodes that are likely to matter, a reduction of over two orders of magnitude.

Formally, each program variant is a pair containing:

1. An *abstract syntax tree* (AST) including all of the statements s in the program.
2. A *weighted path* through that program. The weighted path is a list of pairs $\langle s, w_s \rangle$, each containing a statement in the program visited on the negative test case and the associated weight for that statement.

The default *path weight* of a statement is 1.0 if it is visited in the negative test case but not on any positive test case. Its weight is 0.1 if it is visited on both positive and negative test cases. All other statements have weight 0.0. The weight represents an initial guess of how relevant the statement is to the bug. This approach is related to the union/intersection model of fault localization.²⁰ The *weighted path length* is the sum of statement weights on the weighted path. This scalar gives a rough estimate of the complexity of the search space and is correlated with algorithm performance (Section 4). We return to the issue of fault localization in Section 6.

2.3. Genetic programming

We use GP to maintain a population of program variants. Each variant, sometimes referred to as an *individual*, is represented as an AST annotated with a weighted path (fault localization information). We modify variants using two genetic operations, mutation and crossover. Mutation makes random changes to the nodes along the weighted path, while

crossover exchanges subtrees between two ASTs (see below for details). Each modification produces a new AST and weighted program path. The fitness of each variant is evaluated by compiling the AST and running it on the test cases. Its final fitness is a weighted sum of the positive and negative test cases it passes. Once the fitnesses have been computed for each individual, a *selection phase* deletes the bottom-ranked 50% of the population.^a The new population is formed by first crossing over the remaining high-fitness individuals with the original program. Each crossover produces a single child. We add the children to the population and retain the parents unchanged, maintaining a constant population size. Finally, all surviving individuals are mutated.

The repair process terminates either when it finds a candidate solution that passes all its positive and negative test cases, or when it exceeds a preset number of generations. The first variant to pass all test cases is the *primary repair*.

2.4. Genetic operators

As mentioned above, we apply GP operators to a given variant to produce new program variants, thus exploring the search space of possible repairs. A key operator is *mutation*, which makes random changes to an individual. Because the primitive unit (gene) of our representation is the statement, mutation is more complicated than the simple bit flip used in other evolutionary algorithms. Only statements on the weighted path are subject to the mutation operator. Each location on the weighted path is considered for mutation with probability equal to its path weight multiplied by a *global mutation rate*. A statement selected for mutation is randomly subjected to either *deletion* (the entire statement and all its substatements are deleted: $s \leftarrow \{\}$), *insertion* (another statement is inserted after it: $s \leftarrow \{s; s';\}$), or *swap* of ($s \leftarrow s'$ while $s' \leftarrow s$). Note that a single mutation step in our scheme might contain multiple statement-level mutation operations along the weighted path.

The second operation for manipulating variants is *crossover*, which in GP exchanges subtrees chosen at random between two individuals. Although our initial experiments used a more complicated form of crossover, we have seen that the results do not depend on the particular crossover operator used.⁸ During each generation, every surviving variant undergoes crossover.

Finally, there are a number of other C program components not touched by the GP operators, such as datatype definitions and local and global variable declarations. Because these are never on the weighted path, they are never modified by mutation or crossover. This potentially limits the expressive power of the repairs: If the best fix for a bug is to change a data structure definition, GP will not discover that fix. For example, some programs can be repaired either by reordering the data structure fields, or by changing the program control flow; our technique finds the second repair. Ignoring variable declarations, on the other hand, can cause problems with ill-formed variants. Because of the constraints on mutation and crossover, GP never generates syntactically ill-formed

programs (e.g., it will never generate unbalanced parentheses). However, it could move the use of a variable outside of its declared scope, leading to a semantically ill-formed variant that does not type check and thus does not compile.

2.5. Fitness function

In GP, the *fitness* function is an objective function used to evaluate variants. The fitness of an individual in a program repair task should assess how well the program avoids the program bug while still doing “everything else it is supposed to do.” We use test cases to measure fitness. For our purposes, a *test case* consists of input to the program (e.g., command-line arguments, data files read from the disk, etc.) and an *oracle comparator* function that encodes the desired response. A program P is said to *pass* a test case T iff the oracle is satisfied with the program’s output: $T_{\text{oracle}}(P(T_{\text{input}})) = \text{pass}$. Test cases may check additional behavior beyond pure functional correctness (e.g., the program may be required to produce the correct answer within a given time bound or otherwise avoid infinite loops). Such testing accounts for as much as 45% of total software life-cycle costs,¹⁷ and finding test cases to cover all parts of the program and all required behavior is a difficult but well-studied problem in the field of software engineering.

We call the defect-demonstrating inputs and their anomalous outputs (i.e., the bug we want to fix) the *negative test cases*. We use a subset of the program’s existing test inputs and oracles to encode the core functionalities of the program, and call them the *positive test cases*. Many techniques are available for identifying bugs in programs, both statically (e.g., Ball and Rajamani³ and Hovemeyer and Pugh¹⁰) and dynamically (e.g., Forrest et al.⁷ and Liblit et al.¹³). We assume that a bug has been identified and associated with at least one negative test case.

The fitness function takes a program variant (genotype), compiles the internal representation into an executable program, and runs it against the set of positive and negative test cases. It returns the weighted sum of the test cases passed. The sum is weighted so that passing the negative test cases is worth at least as much as passing the positive test cases. Intuitively, this weighting rewards the search for moving toward a possible repair. Programs that do not compile are assigned fitness zero.

2.6. Minimizing the repair

Because the GP may introduce irrelevant changes, we use program analysis methods to trim unnecessary edits from the primary repair. For example, in addition to the repair, the GP might produce dead code ($\mathbf{x} = 3; \mathbf{x} = 5;$) or calls to irrelevant functions. We use tree-structured difference algorithms and delta debugging techniques in a postprocessing step to generate a simplified patch that, when applied to the original program, causes it to pass all of the test cases.

Using tree-structured differencing,¹ we view the primary repair as a set of changes against the original program. Each *change* is a tree-structured operation such as “take the subtree of the AST rooted at position 4 and move it so that it becomes the 5th child of the node at position 6”. We seek to find a small subset of changes that produces a program that still passes all of the test cases.

Let $C_p = \{c_1, \dots, c_n\}$ be the set of changes associated with

^a We obtained results qualitatively similar to those reported here with a more standard method known as tournament selection.

the primary repair. Let $Test(C) = 1$ if the program obtained by applying the changes in C to the original program passes all positive and negative test cases; let $Test(C) = 0$ otherwise. A *one-minimal subset* $C \subseteq C_p$ is a set such that $Test(C) = 1$ and $\forall c_i \in C. Test(C \setminus \{c_i\}) = 0$. That is, a one-minimal subset produces a program that passes all test cases, but dropping any additional elements causes the program to fail at least one test case. Checking if a set is valid involves a fitness evaluation (a call to *Test*).

We use *delta debugging*²⁵ to efficiently compute a one-minimal subset of changes from the primary repair. Delta debugging is conceptually similar to binary search, but it returns a set instead of a single number. Intuitively, starting with $\{c_1, \dots, c_n\}$, it might check $\{c_1, \dots, c_{n/2}\}$: if that half of the changes is sufficient to pass the *Test*, then $\{c_{1+n/2}, \dots, c_n\}$ can be discarded. When no more subsets of size $n/2$ can be removed, subsets of size $n/4$ are considered for removal, until eventually subsets of size 1 (i.e., individual changes) are tested. Finding the overall minimal valid set by brute force potentially involves $\mathcal{O}(2^n)$ evaluations; delta debugging finds a one-minimal subset in $\mathcal{O}(n^2)$.^{25, Proposition 12} However, we typically observe a linear number of tests in our experiments. This smaller set of changes is presented to the developers as the *final repair* in the form of a standard program patch. In our experiments, the final repair is typically at least an order-of-magnitude smaller than the primary repair.

3. ILLUSTRATION

On 31 December 2008, a bug was widely reported in Microsoft Zune media players, causing them to freeze up.^b The fault was a bug in the following program fragment:^c

```
1 void zunebug(int days) {
2   int year = 1980;
3   while (days > 365) {
4     if (isLeapYear(year)) {
5       if (days > 366) {
6         days -= 366;
7         year += 1;
8       }
9     } else {
10    }
11  }
12  else {
13    days -= 365;
14    year += 1;
15  }
16 }
17 printf("the year is %d\n", year);
18 }
```

^b See *Microsoft Zune affected by "Bug,"* BBC News, December 2008, <http://news.bbc.co.uk/2/hi/technology/7806683.stm>.

^c Downloaded from <http://pastie.org/349916> (Jan. 2009). Note that the original program source code does not make lines 9–10 explicit: our AST represents missing blocks, such as those in *if* statements without *else* clauses, as blocks containing zero statements.

When the value of the input days is the last day of a leap year (such as 10,593, which corresponds to 31 December 2008), the program enters an infinite loop on lines 3–16.

We now walk through the evolution of a repair for this program. We first produce its AST and determine the weighted path, using line numbers to indicate statement IDs. The positive test case zunebug (1,000) visits lines 1–8, 11–18. The negative test case zunebug (10,593) visits lines 1–16, and then repeats lines 3, 4, 8, and 11 infinitely.

For the purposes of this example, the negative test cases consist of the inputs 366 and 10,593, which cause an infinite loop (instead of the correct values, 1980 and 2008), and the positive test cases are the inputs 1,000, 2,000, 3,000, 4,000, and 5,000, which produce the correct outputs 1982, 1985, 1988, 1990, and 1993.

We consider one variant, V , which is initialized to be identical to the original program. In Generation 1, two operations mutate V : the conditional statement "*if* (*days* > 366) {*days* -= 366; *year* += 1;}" is inserted between lines 6 and 7 of the original program; and the statement "*days* -= 366" is inserted between lines 10 and 11. Note that the first insertion includes not just the *if* but its entire subtree. This produces the following code fragment:

```
5  if (days > 366) {
6    days -= 366;
7    if (days > 366) { // insert #1
8      days -= 366;    // insert #1
9      year += 1;      // insert #1
10   }                 // insert #1
11   year += 1;
12 }
13 else {
14 }
15 days -= 366;        // insert #2
```

This modified program passes the negative test case 366 (year 1980) and one positive test case 1000.

Variant V survives Generations 2, 3, 4, 5 unchanged, but in Generation 6, it is mutated with the following operations: lines 6–10 are deleted, and "*days* -= 366" is inserted between lines 13 and 14:

```
5  if (days > 366) {
6    // days -= 366; // delete
7    // if (days > 366) { // delete
8    //   days -= 366;    // delete
9    //   year += 1;      // delete
10   // }                // delete
11   year += 1;
12 }
13 else {
14   days -= 366;        // insert
15 }
16 days -= 366;
```

At this point, V passes all of the test cases, and the search terminates with V as the primary repair. The minimization step is invoked to discard unnecessary changes. Compared to the original program (and using the line numbers from the original), there are three key changes: c_1 = “`days -= 366`” deleted from line 6; c_2 = “`days -= 366`” inserted between lines 9 and 10; and c_3 = “`days -= 366`” inserted between lines 10 and 11. Only c_1 and c_3 are necessary to pass all tests, so change c_2 is deleted, producing the final repair:

```

1 void zunebug_repair (int days) {
2     int year = 1980;
3     while (days > 365) {
4         if (isLeapYear (year)) {
5             if (days > 366) {
6                 // days -= 366; // deleted
7                 year += 1;
8             }
9             else {
10            }
11            days -= 366; // inserted
12        } else {
13            days -= 365;
14            year += 1;
15        }
16    }
17    printf ("the year is %dn", year);
18 }

```

On average, constructing and minimizing a repair for the Zune fragment shown here takes our prototype a total 42 s, including the time to compile and evaluate variants against a suite of five positive and two negative tests.

4. RESULTS

To date, we have repaired 20 defects in modules totaling 186kLOC from 20 programs totaling 2.3MLOC (not all shown here). We have repaired defects from eight classes: infinite loop, segmentation fault, heap buffer overrun, nonoverflow denial of service, integer overflow, invalid exception, incorrect output, and format string vulnerability. Constructing a repair requires 1428 s on average, most of which is spent performing an average of 3903 fitness evaluations. In Table 1, we summarize results for 11 benchmark programs reported in Forrest et al.⁸ and Weimer et al.²⁴ The benchmark programs, test cases, GP code, and the supporting infrastructure used to generate and reproduce these results are available at: <http://genprog.adaptive.cs.unm.edu/>.

In all of our experiments, a standard *trial* uses the following setup. The population size is 40, and GP runs for a maximum of 20 generations. For the first 10 generations, the global mutation rate is 0.06. If no primary repair is found, the current population is discarded, the global mutation rate is halved to 0.03, and, if possible, the weighted path is restricted to statements visited solely during the negative test case, and the GP is run for 10 additional generations. These results show that GP can automatically discover repairs for a variety of documented bugs in production C programs.

Table 1. Eleven defects repaired by Genetic Programming, summarized from previous work. The size of each program is given in lines of code as well as weighted path units (see Section 2.2). Each repair used five or six positive tests and one or two negative tests. The “Time” column gives the total wall-clock time required to produce and minimize the repair (on a successful trial). The “Fitness Evals” column lists the number of times the entire fitness function was called before a repair was found (averaged over only the successful trials). The “Repair Size” column gives the size of the final minimized repair, as measured in lines by the Unix `diff` utility.

Program	Lines of Code	Weighted Path	Description	Fault	Time (s)	Fitness Evals	Repair Size
gcd	22	1.3	Euclid’s algorithm	Infinite loop	153	45.0	2
zune	28	2.9	MS Zune excerpt	Infinite loop	42	203.5	4
uniq utx 4.3	1146	81.5	Duplicate filtering	Segmentation fault	34	15.5	4
look utx 4.3	1169	213.0	Dictionary lookup	Segmentation fault	45	20.1	11
look svr 4.0	1363	32.4	Dictionary lookup	Infinite loop	55	13.5	3
units svr 4.0	1504	2159.7	Metric conversion	Segmentation fault	109	61.7	4
deroff utx 4.3	2236	251.4	Document processing	Segmentation fault	131	28.6	3
nullhttpd 0.5.0	5575	768.5	Webserver	Heap buffer overrun	578	95.1	5
indent 1.9.1	9906	1435.9	Source code formatting	Infinite loop	546	108.6	2
flex 2.5.4a	18775	3836.6	Lexical analyzer generator	Segmentation fault	230	39.4	3
atris 1.0.6	21553	34.0	Graphical tetris game	Stack buffer overrun	80	20.2	3
Total	63277	8817.2			2003	651.2	39

The trial terminates if it discovers a primary repair. We performed 100 trials for each program, memoizing fitnesses such that within one trial, two individuals with different ASTs but the same source code are not evaluated twice. Similarly, individuals that are copied to the next generation without change are not reevaluated.

Once a primary repair has been located, the process of minimizing it to a final repair is quite rapid, requiring less than 5 s on average. Final repairs, expressed in **patch** format, varied in size from four lines (e.g., **zune**: one insert, one delete, and one context-location line for each edit) to 11 lines (e.g., **look utx** 4.3).

Not all trials lead to a successful repair; in the repairs shown here, an average of 60% of trials produce a primary repair. The “Time” and “Fitness Evals” columns in Table 1 measure the effort taken for a successful trial. Since all trials are independent, a number of trials can be run in parallel, terminating when the first successful trial yields a primary repair. In addition, the fitnesses of different variants and the results of different test cases for a given variant can all be evaluated independently, making our approach easy to parallelize on multicore architectures. The measurement in Table 1 was made on a quad-core 3 GHz machine.

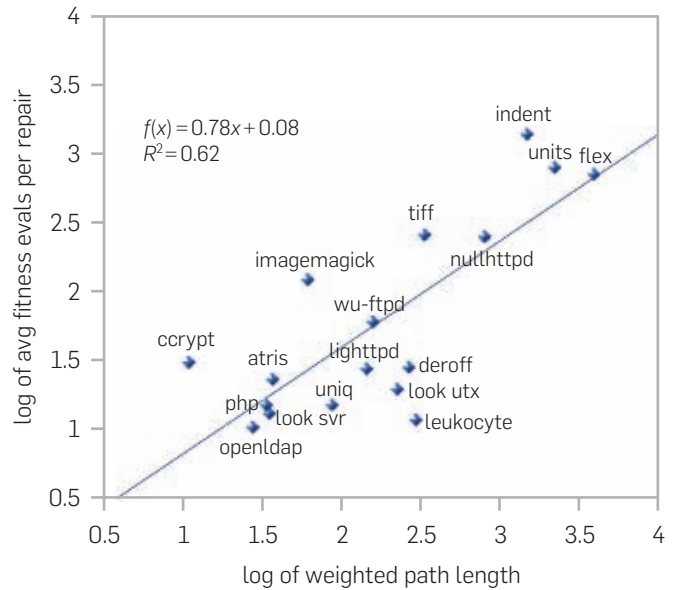
Over half of the total time required to create a repair is spent evaluating fitness by running compiled variants through test cases. For programs with large test suites, this cost can be considerable. A dominant factor in the scalability of our approach is thus the number of such fitness evaluations that must be made to find a repair. The number of fitness evaluations required is related to the size of the search space and the efficacy of the search strategy: each fitness evaluation represents a single probe. We hypothesize that the size of the weighted path is a good representation of the search space size; recall that we only modify statements along the path (Sections 2.2 and 2.4). Figure 1 shows the results of an empirical investigation into this relationship, plotting the average number of fitness evaluations required to produce each of 18 repairs against the length of their weighted paths (note log–log scale). Although more data points are needed before strong conclusions can be drawn, the plot suggests that the number of fitness evaluations, and thus the search time, may scale as a power law of the form $y = ax^b$ where b is the slope of the best fit line (0.78). This suggests that the time to find a repair scales nearly linearly with the size of the weighted path for a fixed number of test cases.

Our approach has also failed to repair some defects, including those that require many simultaneous edits or changes that cannot be made directly at the statement level (e.g., **matmul(b, a)** should be **matmul(a, b)**). We return to the issue of repair quality in Section 6.

5. RELATED WORK

Arcuri² proposed the idea of using GP to repair software bugs automatically, and Orlov and Sipper experimented with evolving Java bytecode.¹⁵ However, our work is the first to report substantial experimental results on real programs with real bugs. The field of Search-Based Software Engineering (SBSE)⁹ uses evolutionary and related methods for software testing, e.g., to develop test suites, improve

Figure 1. GP search time scales with weighted path size. Data are shown for 18 programs successfully repaired by GP (gcd and zune examples omitted; figure includes several additional programs to those listed in Table 1), with best linear fit. The x-axis is the base-10 logarithm of the weighted path length, and the y-axis shows the logarithm of the total number of fitness evaluations performed before the primary repair is found (averaged over 100 runs).



software project management, and effort estimation find safety violations and in some cases refactor or reengineer large software bases. In SBSE, most innovations in the GP technique involve new kinds of fitness functions, and there has been less emphasis on novel representations and operators, such as those explored here.

Our approach automatically repairs programs without specifications. In previous work, we developed an automatic algorithm for soundly repairing programs with specifications.²³ However, formal specifications are not always available (e.g., there were no formal specifications available for any of the programs repaired here), so the present work focuses on test cases to check and ensure correctness.

Trace and fault localization, minimization, and explanation (e.g., Jones and Harrold¹¹) projects also aim to elucidate faults and ease debugging. These approaches typically narrow down an error symptom to a few lines (a potential cause). Our work extends this work by proposing a concrete repair. In addition, these other algorithms are usually limited to the given trace or source code and will thus never localize the “cause” of an error to a missing statement or suggest that a statement be moved. Our approach can infer new code that should be added, deleted, or swapped: 6 of the 11 repairs in Table 1 required insertions or swaps.

Demsky et al.⁵ present a technique for data structure repair. Given a formal specification of data structure consistency, they modify a program so that if the data structures ever become inconsistent, they can be modified back to a consistent state at runtime. Their technique does not modify the program source code in a user-visible way. Instead, it inserts runtime monitoring code that “patches

up” inconsistent state so that the buggy program can continue to execute. Thus, their programs continue to incur the runtime overhead after the repair is effected. Another difference from our work is that their data structure repair requires formal specifications. Finally, their technique is limited to data structures and does not address the full range of logic errors. The `gcd` infinite loop in Section 3, for example, is outside the scope of this technique. However, this technique complements ours: in cases where runtime data structure repair does not provide a viable long-term solution, it may enable the program to continue to execute while our technique searches for a long-term repair.

Clearview¹⁶ automatically detects and patches assembly-level errors in deployed software. Clearview monitors a program at runtime, learns invariants that characterize normal behavior, and subsequently flags violations for repair. Candidate patches that make the implicated invariant true are generated and tested dynamically. Although the performance overhead of Clearview is high, it has successfully been applied to buggy versions of Mozilla Firefox and evaluated against a Red Team of hackers. However, Clearview can repair only those errors that are relevant to selected monitors. Our method is more generic, providing a single approach to repair multiple classes of faults without the need for specific monitors, and we do not require continual runtime monitoring (and the incumbent slowdown) to create and deploy repairs.

This body of work illustrates a burgeoning interest in the problem of automated software repair and some of the many possible approaches that might be tried. There are several other recent but less mature proposals for automatically finding and repairing bugs in software, e.g., Dallmeier et al.,⁴ suggesting that we can expect rapid progress in this area over the next several years.

6. DISCUSSION

The results reported here demonstrate that GP can be applied to the problem of bug repair in legacy C programs. However, there are some caveats.

Basic limitations. First, we assume that the defect is reproducible and that the program behaves deterministically on the test cases. This limitation can be mitigated by running the test cases multiple times, but ultimately if the program behavior is random it will be difficult for our method to find or evaluate a correct patch. We further assume that positive test cases can encode program requirements. Test cases are much easier to obtain than formal specifications or code annotations, but if too few are used, a repair could sacrifice important functionality. In practice, we are likely to have too many test cases rather than too few, slowing down fitness evaluation and impeding the search. We also assume that the path taken along the negative test case is different from the positive path. If they overlap completely, our weighted representation will not guide GP modifications as effectively. Finally, we assume that the repair can be constructed from statements already extant in the program; in future work, we plan to extend our method to include a library of repair templates.

Evolution. One concern about our results to date is the role of evolution. Most of our repairs result from one or two random modifications to the program, and they are often found

within the first few generations or occasionally, not at all. We have conducted some experiments using a brute force algorithm (which applies simple mutation operations in a predetermined order) and random search (which applies mutation operations randomly without any selection or inheritance of partial solutions). Both these simpler alternatives perform as well or better than the GP on many, but not all, of our benchmark programs. We do not fully understand what characteristics, either of the program or the particular bug, determine how easily a solution can be found through random trial and error. However, thus far GP outperforms the other two search strategies in cases where the weighted path is long (i.e., where the fault is difficult to localize). There are several interesting questions related to the design of our GP algorithm, but the overall process proved so successful initially that we have not experimented carefully with parameter values, selection strategies, and operator design. These all could almost certainly be improved.

Fault localization. As Figure 1 shows, the time to find a solution varies with the length of the weighted path. Since the weighted path is a form of fault localization, we could use off-the-shelf fault localization techniques (e.g., Jones and Harrold¹¹ and Renieres and Reiss²⁰) or dynamically discovered invariants,¹³ in the style of Daikon⁶ to further narrow the search space. Predicates over data might help in cases where faults cannot be localized by control flow alone, such as cross-site scripting or SQL injection attacks. In addition, our recent experiments have shown that the location of the fault (i.e., where to insert new code) is rarely the same as source of the fix (i.e., where to find code to insert). Since more than half of our repairs involve inserting or swapping code, locating viable fixes is of critical importance but remains poorly understood.

Fitness function. Our current test suite fitness function has the advantage of conceptual simplicity: a variant that passes all test cases is assumed to be correct, and a variant that does not compile or fails all tests is rejected. However, it may not be accurate in the middle ranges or precise enough to guide the evolutionary search in more complex problems. Consider a program with a race condition, for which the fix consists of inserting separate `lock` and `unlock` calls. A variant with a partial solution (e.g., just an inserted `lock`) may unfortunately pass fewer test cases (e.g., by deadlocking), thus “deceiving” the evolutionary algorithm. Fitness function design could be enhanced in several ways, for example, by weighting individual test cases, dynamically choosing subsets of test cases to be included, or by augmenting the test case evaluation with other information. For example, if a simple predicate like `x == y` is true at a particular point on all positive test cases, but false for the negative test, a variant that causes it to be true for the negative test might be given a higher fitness value.

Repair quality. We are interested in how much repairs vary after minimization, and how repair quality compares to human-engineered solutions. In our experiments to date, many, but not all, repairs look identical after the minimization step. For example, we have isolated 12 distinct repairs for the `null-httpd` fault, but much overlap exists (e.g., two repairs may insert the same statement into different points in the same basic block). Repair quality depends on the presence of a

high-quality set of positive test cases that encode program requirements. In other work, we experimented with held-out indicative workloads, fuzz testing, and held-out exploits to demonstrate that our server repairs address the causes of problems without being fragile memorizations of the negative input and without failing common requests (i.e., because of the positive tests). Much remains to be done in this area, however, such as automatically documenting or proving properties of the generated repairs.

Future work. Beyond these immediate steps, there are other more ambitious areas for future work. For example, we plan to develop a generic set of repair templates so the GP has an additional source of new code to use in mutation, beyond those statements that happen to be in the program. Our AST program representation could be extended in various ways, for example, by including data structure definitions and variable declarations. Similarly, we are currently experimenting with assembly- and bytecode-level repairs. Finally, we are interested in testing the method on more sophisticated errors, such as race conditions, and in learning more about bugs that need to be repaired, such as their size and distribution, and how we might identify which ones are good candidates for the GP technique.

7. CONCLUSION

We credit much of the success of this technique to design decisions that limit the search space. Restricting attention to statements, focusing genetic operations along the weighted path, reusing existing statements rather than inventing new ones, and repairing existing programs rather than creating new ones, all help to make automatic repair of errors using GP tractable in practice.

The dream of automatic programming has eluded computer scientists for at least 50 years. The methods described in this paper do not fulfill that dream by evolving new programs from scratch. However, they do show how to evolve legacy software in a limited setting, providing at least a small down payment on the dream. We believe that our success in evolving automatic repairs may say as much about the state of today's software as it says about the efficacy of our method. In modern environments, it is exceedingly difficult to understand an entire software package, test it adequately, or localize the source of an error. In this context, it should not be surprising that human programming often has a large trial and error component, and that many bugs can be repaired by copying code from another location and pasting it in to another. Such debugging is not so different from the approach we have described in this paper.

Acknowledgments

We thank David E. Evans, Mark Harman, John C. Knight, Anh Nguyen-Tuong, and Martin Rinard for insightful discussions. This work is directly based on the seminal ideas of John Holland and John Koza.

This research was supported in part by National Science Foundation Grants CCF 0621900, CCR-0331580, CNS 0627523, and CNS 0716478, Air Force Office of Scientific Research grant FA9550-07-1-0532, as well as gifts from Microsoft Research. No official endorsement should be inferred. The authors thank Cris Moore for help finding the Zune code. 

References

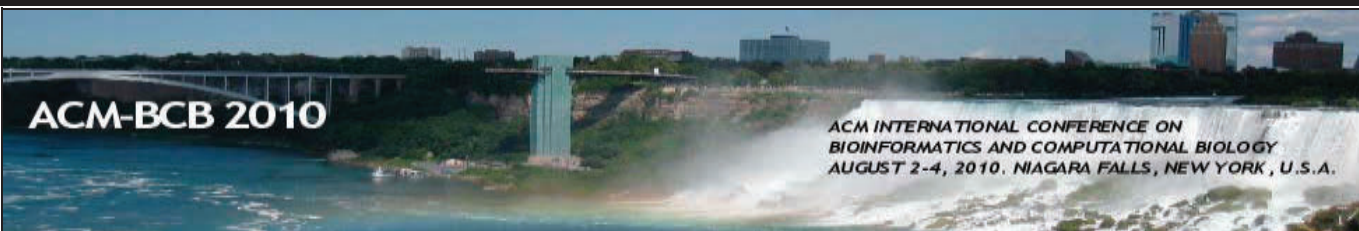
- Al-Ekram, R., Adma, A., Baysal, O. diffX: an algorithm to detect changes in multi-version XML documents. In *Conference of the Centre for Advanced Studies on Collaborative Research*. IBM Press, 2005, 1–11.
- Arcuri, A., Yao, X. A novel co-evolutionary approach to automatic software bug fixing. In *IEEE Congress on Evolutionary Computation* (2008), 162–168.
- Ball, T., Rajamani, S.K. Automatically validating temporal safety properties of interfaces. In *SPIN Workshop on Model Checking of Software* (May 2001), 103–122.
- Dallmeier, V., Zeller, A., Meyer, B. Generating fixes from object behavior anomalies. In *International Conference on Automated Software Engineering* (2009).
- Demsky, B., Ernst, M.D., Guo, P.J., McCamant, S., Perkins, J.H., Rinard, M. Inference and enforcement of data structure consistency specifications. In *International Symposium on Software Testing and Analysis* (2006), 233–244.
- Ernst, M.D., Perkins, J.H., Guo, P.J., McCamant, S., Pacheco, C., Tschantz, M.S., Xiao, C. The Daikon system for dynamic detection of likely invariants. *Sci. Comput. Program* (2007).
- Forrest, S., Hofmeyr, S.A., Somayaji, A., Longstaff, T.A. A sense of self for Unix processes. In *IEEE Symposium on Security and Privacy* (1996), 120–128.
- Forrest, S., Weimer, W., Nguyen, T., Le Goues, C. A genetic programming approach to automated software repair. In *Genetic and Evolutionary Computing Conference* (2009), 947–954.
- Harman, M. The current state and future of search based software engineering. In *International Conference on Software Engineering* (2007), 342–357.
- Hovemeyer, D., Pugh, W. Finding bugs is easy. In *Object-Oriented Programming Systems, Languages, and Applications Companion* (2004), 132–136.
- Jones, J.A., Harrold, M.J. Empirical evaluation of the tarantula automatic fault-localization technique. In *International Conference on Automated Software Engineering* (2005), 273–282.
- Koza, J.R. *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.
- Liblit, B., Aiken, A., Zheng, A.X., Jordan, M.I. Bug isolation via remote program sampling. In *Programming Language Design and Implementation* (2003), 141–154.
- Necula, G.C., McPeak, S., Rahul, S.P., Weimer, W. Cil: An infrastructure for C program analysis and transformation. In *International Conference on Compiler Construction* (Apr. 2002), 213–228.
- Orlov, M., Sipper, M. Genetic programming in the wild: Evolving unrestricted bytecode. In *Genetic and Evolutionary Computation Conference* (ACM, 2009), 1043–1050.
- Perkins, J.H., Kim, S., Larsen, S., Amarasinghe, S., Bachrach, J., Carbin, M., Pacheco, C., Sherwood, F., Sidiropoulos, S., Sullivan, G., Wong, W.-F., Zibin, Y., Ernst, M.D., Rinard, M. Automatically patching errors in deployed software. In *ACM Symposium on Operating Systems Principles* (Oct. 2009), 87–102.
- Pigoski, T.M. *Practical Software Maintenance: Best Practices for Managing Your Software Investment*. John Wiley & Sons, Inc., 1996.
- Polli, R., Langdon, W.B., McPhee, N.F. *A Field Guide to Genetic Programming*. Published via <http://lulu.com> and freely available at <http://www.gp-field-guide.org.uk>, 2008.
- Ramamoorthy, C.V., Tsai, W.-T. Advances in software engineering. *IEEE Comput.* 29, 10 (1996), 47–58.
- Renieres, M., Reiss, S.P. Fault localization with nearest neighbor queries. In *International Conference on Automated Software Engineering* (Oct. 2003), 30–39.
- Schmidt, M., Lipson, H. Distilling free-form natural laws from experimental data. *Science* 324, 5923 (2009), 81–85.
- Seacord, R.C., Plakosh, D., Lewis, G.A. *Modernizing Legacy Systems: Software Technologies, Engineering Process and Business Practices*. Addison-Wesley, 2003.
- Weimer, W. Patches as better bug reports. In *Generative Programming and Component Engineering* (2006), 181–190.
- Weimer, W., Nguyen, T., Le Goues, C., Forrest, S. Automatically finding patches using genetic programming. In *International Conference on Software Engineering* (2009), 364–367.
- Zeller, A., Hildebrandt, R. Simplifying and isolating failure-inducing input. *IEEE Trans. Software Eng.* 28, 2 (2002), 183–200.

Westley Weimer (weimer@virginia.edu), University of Virginia.

Stephanie Forrest (forrest@cs.unm.edu), University of New Mexico.

Claire Le Goues (legoues@virginia.edu), University of Virginia.

ThanhVu Nguyen (tnguyen@cs.unm.edu), University of New Mexico.



ACM-BCB 2010

ACM INTERNATIONAL CONFERENCE ON
BIOINFORMATICS AND COMPUTATIONAL BIOLOGY
AUGUST 2-4, 2010, NIAGARA FALLS, NEW YORK, U.S.A.

Call for Papers and Posters

ACM International Conference On Bioinformatics and Computational Biology (ACM-BCB 2010)

August 2-4, Niagara Falls, New York, USA

ACM-BCB Website: <http://www.cse.buffalo.edu/ACM-BCB2010>

Key Dates

Paper Submission	Mar. 15, 2010
Poster Submission	Apr. 11, 2010
Notification to Authors	May 1, 2010

Organizing Committee

Steering Committee

Joydeep Ghosh, University of Texas at Austin
Vipin Kumar, University of Minnesota
Meral Ozsoyoglu, Case Western Reserve University
Yi Pan, Georgia State University
Dong Xu, University of Missouri-Columbia
Aidong Zhang, SUNY at Buffalo (co-chair)
Joe Zhang, University of Southern Mississippi (co-chair)

General Chairs

Aidong Zhang, SUNY at Buffalo
Mark Borodovsky, Georgia Tech

Program Chairs

Gultekin Ozsoyoglu, Case Western Reserve University
Armin Mikler, University of North Texas

Treasurer

Joe Zhang, University of Southern Mississippi

Workshop Chairs

Michael Buck, SUNY at Buffalo
Vasant Honavar, Iowa State University
Zhongming Zhao, Vanderbilt University Medical Center
Jason Wang, New Jersey Institute of Technology

Poster Chairs

Tony Hu, Drexel University
Sun Kim, Indiana University

Publicity Chairs

Mary Qu Yang, NIH
Kebin Jia, Beijing University of Technology

Student Awards Chairs

Susan Bridges, Mississippi State University
Parthasarathy Srinivasan, Ohio State University

Best Paper Awards Chairs

Joydeep Ghosh, University of Texas at Austin
Wei Wang, University of North Carolina at Chapel Hill

Panel Chair

Vipin Kumar, University of Minnesota

Keynote Chair

Dong Xu, University of Missouri-Columbia

Proceedings Chairs

Li Liao, University of Delaware
Guo-zheng Li, Tongji University

Tutorials Chairs

Mohammed Zaki, Rensselaer Polytechnic Institute
Bo Yuan, Shanghai Jiaotong University

Registrations Chair

Preetam Ghosh, University of Southern Mississippi

Local Arrangements

Zihua Hu, SUNY at Buffalo
Marianne Sullivan, SUNY at Buffalo

ACM-BCB is the first ACM conference in the areas of bioinformatics, computational biology, and biomedical researches. The conference is pleased to announce that the conference has set up three awards: the best paper award, the best student paper award, and the best poster award. The conference will also support 10 student travel grants (see the conference website for more details).

We invite you to submit papers with unpublished, original research describing recent advances on all aspects of Bioinformatics and Computational Biology. All papers will undergo peer review by the conference program committee; accepted papers will be published in the conference proceedings (in CD) and in the ACM Digital Library. Authors of selected papers will be invited to extend their papers for submission to several journals.

Papers should not exceed 10 pages in ACM template on 8.5 x 11 inch paper (see ACM [templates](http://www.acm.org/signs/publications/proceedings-templates) at <http://www.acm.org/signs/publications/proceedings-templates>).

Papers should be submitted via the link

(<https://cmt.research.microsoft.com/BCB2010/>).

We also invite you to submit posters focusing on problems and new challenges of Bioinformatics and Computational Biology. Submit your posters (limited to 3 pages of ACM template) to poster co-chair: Tony Hu (xh29@drexel.edu). Accepted posters will be published in the conference proceedings.

Topics

Assembly algorithms for the next generation sequencing

Genome annotations via mass spectrometry

Gene discovery and protein function prediction

Functional genomics algorithms for high throughput data generation methods

DNA and protein motif discovery

Models of gene expression regulation at RNA level

Algorithms for comparative and evolutionary genomics

Molecular evolution and phylogeny reconstruction

Multiple genome alignment

Computational genomics for plant and animal genomes

Engineering of bioinformatic databases

Models of gene expression networks

Computational proteomics and metabolomics

Prediction of protein-protein interactions

Protein structure prediction and molecular dynamics

Visualization of biomolecular and cellular structure, dynamics and interactions

Modeling of metabolic pathways

Computational epigenomics

Models of cellular and multicellular systems

Prediction of cellular network dynamics from incomplete information

Clinical genomics and proteomics

Data mining in biomolecular databases

High performance computing for complex biomolecular and cellular systems

Biomedical imaging



Association for
Computing Machinery

Advancing Computing as a Science & Profession



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
University at Buffalo The State University of New York



National Science Foundation
WHERE DISCOVERIES BEGIN

CAREERS

Case Western Reserve University Tenure Track Assistant Professor Positions In Biomedical Informatics

The Division for Medical Informatics in Case Western Reserve University's School of Medicine invites applications for two tenure track Assistant Professor positions. We are seeking candidates whose research areas intersect computer science and clinical and health science, with an aptitude to develop software tools, methodologies and informatics infrastructure with a direct bearing on the biomedical research enterprise. Specific areas of interest include, but are not limited to (in the context of clinical, comparative and translational research):

- Knowledge assimilation and intelligent decision support;
- Biomedical ontologies and their applications;
- Machine learning and natural language processing;
- Data collection, storage, management, integration, query, and mining.

Must have a Ph.D. or equivalent in computer science or biomedical informatics. Minimum 5 years experience in biomedical informatics, computer science or health IT.

Applicants should provide via email (JXA136@case.edu) a current CV, a statement (2-3 pages) of their research plans, and the names/contact info of four references. Specify Search #204/205 in the email subject line. Enquiries about the positions may be directed to Professor GQ Zhang, Division Chief, at GQ@case.edu.

Case Western Reserve, a leader among independent research universities, offers world-class academic programs in Medicine and Engineering. In employment, as in education, Case Western Reserve University is committed to Equal Opportunity and Diversity. Women and under-represented minorities are encouraged to apply.

DOCOMO Communications Laboratories USA, Inc. Research Engineer (Full-Time/ Contractor/Intern)

DOCOMO USA Labs is looking for a Ph.D. level researcher to work in data analysis, distributed computing, and optimization. The candidate must have a track record of publications in top tier conferences and/or journals. Strong background in the following areas is required: statistical algorithm design, signal processing, graph theory, queuing theory, stochastic optimization, control theory, game theory, and information theory. Strong programming experience in Java and statistical programming languages like R are required. Experience with implementation on distributed computing platforms like MapReduce or Hadoop on Linux systems will be a plus.

Applicants must have the relevant authorization to work in a U.S. company.

Please indicate Code: OOP-RE-ACM, Email Address: oop_recruit@docomolabs-usa.com. Apply URL: <http://www.docomolabs-usa.com/>

Drexel University University Professorships

Drexel University (drexel.edu) invites applications for anticipated senior faculty positions (University Professors) in key science and engineering areas aligned with strategic university-wide research initiatives. Hiring and appointments will be directed by the Office of the Provost in collaboration with relevant colleges/schools and academic departments across the University and the possibility of interdisciplinary appointments.

Drexel University is in the midst of a period of unprecedented growth and opportunity. Drexel seeks to identify and recruit senior faculty committed to advancing current interdisciplinary research programs and creating new ones. More information is available in the University's current strategic plan: <http://www.drexel.edu/provost/strategicplan/>

Key areas of strategic interest for which Drexel University is currently searching include:

- Verification and validation of complex software systems
- Modeling & simulation (e.g., computer networks, air space decongestion, cyber-physical systems, etc)
- CyberSecurity, CyberWarfare, CyberForensics, Information Assurance

Drexel seeks individuals who have demonstrated skills in working across college and departmental boundaries and are experienced in building and managing large-scale, interdisciplinary, research and educational efforts.

Successful applicants will have a distinguished track record of leadership, research execution and management, as well as teaching and service. Interested candidates should apply online at:

[www.drexeljobs.com/applicants/
Central?quickFind=73467](http://www.drexeljobs.com/applicants/Central?quickFind=73467)

Please complete the short online application and submit a cover letter, CV, brief statements describing your research program and teaching philosophy, and contact information for at least four references. Review of applications will commence immediately.

Drexel University is an Affirmative Action/Equal Opportunity Employer.

Georgia State University Department of Computer Science Assistant or Associate Professor

The Department of Computer Science of Georgia State University invites applications for an anticipated tenure-track position for Assistant or Associate Professor beginning the Fall semester, 2010, pending budgetary approval. Earned Ph.D. in Computer Science, or a closely related discipline, and excellent records of publications and funding are required. Preference is for individual with specialty in distributed systems and networking. An offer of employment will be conditional on background verification.

Georgia State University, founded in 1913, is a Carnegie Doctoral/Research Extensive university. Located in the heart of downtown Atlanta, this major research university has an enrollment of more than 30,000 undergraduate and graduate students in six colleges. Georgia State is the second largest university in the state, with students coming from every county in Georgia, every state in the nation and from over 145 countries. Georgia State University is currently embarking on a record \$1 billion campus expansion. The Computer Science Department offers programs leading to the B.S., M.S., and Ph.D. degrees in computer science. Currently, more than 60 PhD students are enrolled. Departmental computing facilities for research and instruction include a departmental network of PCs, Unix/Linux workstations, two interconnected Beowulf clusters, and a 24-processor shared-memory high-performance computer, and five labs, including a hypermedia and visualization research laboratory. A full-time systems programmer supports the departmental computing facilities. All four of the department's assistant professors currently hold awards from the National Science Foundation's Faculty Early Career Development (CAREER) Program.

Applicants should send: letter of interest, C.V. without birth date, but with citizenship status, and three letters of recommendation and transcripts of all graduate work to:

Dr. Yi Pan, Chair
Department of Computer Science
Georgia State University
34 Peachtree Street, Suite 1450
Atlanta, Georgia, 30303

Applications can also be sent via email to pan@cs.gsu.edu and will be accepted until position is filled.

Georgia State University, a Research University of the University System of Georgia, is an AA/EEO employer.

Goshen College Informatics Professor

Develop and teach courses in Informatics, advise student capstone projects, work with other departments to develop cognates, and promote the new program. PhD preferred, Master's degree required, in a computing field. See www.goshen.edu/employment for more information.

Institute for Infocomm Research (I²R) Research Scientist Human Factors Engineering Ref: I2R /R/300110/04

You will develop and contribute to the development of new human factors methods across a variety of hardware and software platforms, and domains. You will provide thought leadership in research and development of ideas, and translate the research into intellectual properties and publications. You will also work with other technologists and re-

searchers to understand user requirements in their social and physical context and guide more junior human factors researchers to generate ideas.

Requirements:

- A PhD in Human Factors Engineering, Human-Computer Interaction or other research-based behavioral science discipline.
- At least 5 years of experience in applying Human Factors/ Usability evaluation techniques, preferably in infocomm, consumer electronics or healthcare domains.
- Good communication and writing skills with an excellent record of publications and/or success in projects.
- Ability to work independently as well as in teams and willing to take up challenges.
- Strong R&D leadership and ability to build up new competency in the areas mentioned above.

For interested candidate, please send your resume to joinus@i2r.a-star.edu.sg. Alternatively please logon to www.i2r.a-star.edu.sg, for more career opportunities.

HIIT Helsinki Institute for Information Technology Director

Aalto University and the University of Helsinki are looking for a proven leader to be the next Director of Helsinki Institute for Information Technology HIIT.

The new leader will create and execute an ambitious research agenda in an institute conducting basic and applied research in ICT. With almost 200 multidisciplinary researchers, a budget

of EUR 10 million and partnerships with global ICT corporations and major research institutions the Director has an opportunity to contribute to ICT research with world-class results.

The parent universities of HIIT offer up to a five years contract starting from 1 August 2010 and excellent facilities in Finland. The position is renewable. The application deadline is 19 March 2010 at 3.45 pm Finnish time.

More information at www.hiit.fi/jobs

Pacific Northwest National Laboratory Senior Research Scientist in Data Intensive Scientific Computing

PNNL is embarking on an ambitious new R&D agenda to create the next generation of data management and analysis technologies for large scale scientific challenges. We are looking for a senior computer scientist to help drive this agenda. The position involves all aspects of defining, creating and deploying complex, innovative software tools that leverage Cloud- and HPC-based computational platforms. Job 118396. Apply URL: <http://jobs.pnl.gov>

Professionals Incorporated Software Engineer

Multiple Software Engineer openings in scenic Ithaca, NY situated at the base of Cayuga Lake and home of wineries, Cornell University and more.

BS in Computer Science from an accredited University; 2-20 years of exp; C/C++. Interested in Compilers, Scheme, and/or Python. US Citizens Only. Relocation available! Apply URL: <http://www.cpsprofessionals.com>

University of Campinas (Unicamp) Assistant Professor

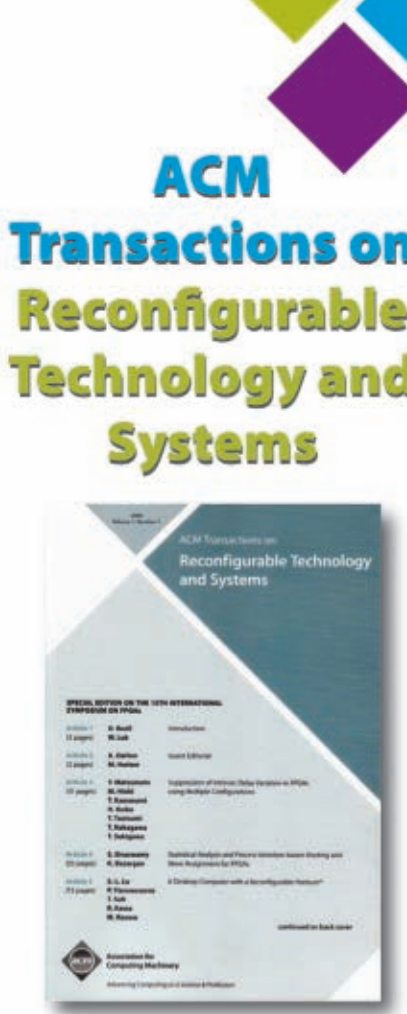
The University of Campinas (Unicamp), one of the leading research universities in Brazil, is seeking applications from Ph.D.-holders with high academic achievement records in computer science and strong motivation to pursue quality research. We will provide good working conditions at the Institute of Computing (IC), compatible salary, well-fare, tenure-track positions, and good career development perspectives. Brazil is a fast developing country. Be part of the progress. Come work with us.

Successful candidates must hold a Ph.D. degree in computer science/engineering or a closely related field; have strong commitment to academic excellence and teaching; and be interested in establishing and leading an independent and creative research group. If you would like to join us, please send a CV in English or Portuguese to carreiras@reitoria.unicamp.br. Send a message to info.carreiras@reitoria.unicamp.br if you need more information.

University of Colorado at Colorado Springs Assistant Professor

The Computer Science Department at the University of Colorado at Colorado

Spring invites applications at Assistant Professor level with research focus on cyber security. <http://cs.uccs.edu/~cyber/>



ACM

Transactions on Reconfigurable Technology and Systems

◆ ◆ ◆ ◆ ◆

This quarterly publication is a peer-reviewed and archival journal that covers reconfigurable technology, systems, and applications on reconfigurable computers. Topics include all levels of reconfigurable system abstractions and all aspects of reconfigurable technology including platforms, programming environments and application successes.

◆ ◆ ◆ ◆ ◆

www.acm.org/trets
www.acm.org/subscribe



Association for
Computing Machinery



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 ACM guidelines. NO PROOFS can be sent. Classified line ads are NOT commissionable.

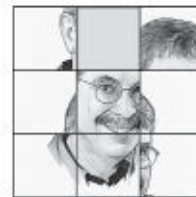
Rates: \$325.00 for six lines of text, 40 characters per line. \$32.50 for each additional line after the first six. The MINIMUM is six lines.

Deadlines: Five weeks prior to the publication date of the issue (which is the first of every month). Latest deadlines: <http://www.acm.org/publications>

Career Opportunities Online: Classified and recruitment display ads receive a free duplicate listing on our website at: <http://campus.acm.org/careercenter>

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



DOI:10.1145/1735223.1735250

Peter Winkler

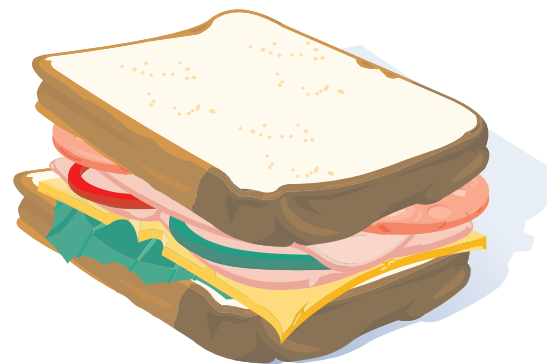
Puzzled

Variations on the Ham Sandwich Theorem

Welcome to three new puzzles. Solutions to the first two will be published next month; the third is (as yet) unsolved. In each, the issue is how your intuition matches up with the mathematics.

The solutions of the first two puzzles—and maybe the third as well—make good use of the Intermediate Value Theorem, which says if you go continuously from one real number to another, you must pass through all the real numbers in between. The most famous application is perhaps the Ham Sandwich Theorem, which says, given any ham-and-cheese sandwich, no matter how sloppily made, there is a planar cut dividing the ham, cheese, and bread, each into two equal-size portions.

The two solved problems are perhaps a bit easier than the Ham Sandwich Theorem but still tricky and rewarding enough to be worth your attention and effort.



1. A pair of intrepid computer programmers spend a weekend hiking the Cascade Range in Washington. On Saturday morning they begin an ascent of Mt. Baker—all 10,781 feet of it—reaching the summit by nightfall. They spend the night there and start down the mountain the following morning, reaching the bottom at dusk on Tuesday.

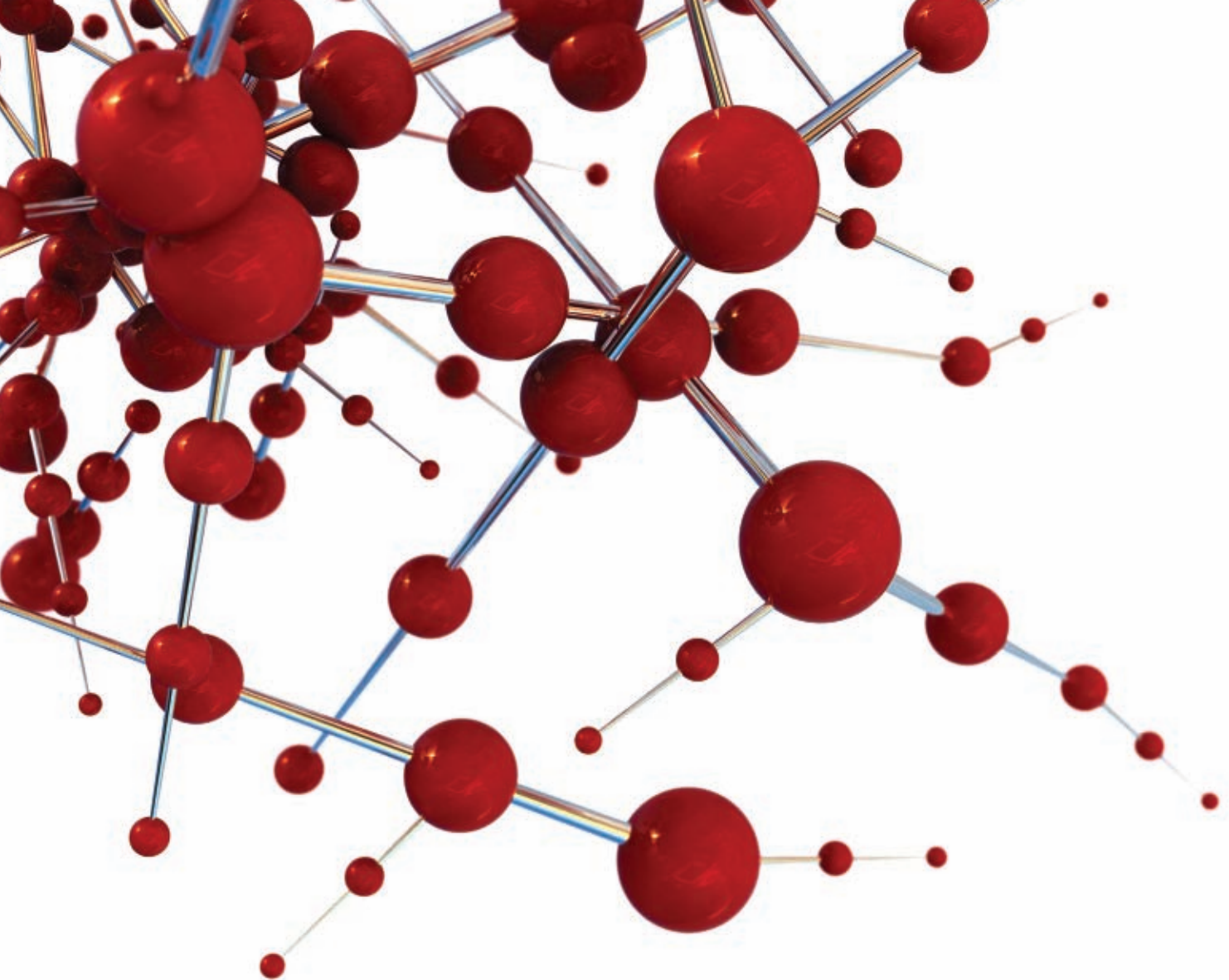
Prove that at some precise time of day, these programmers were at exactly the same altitude on Sunday as they were on Saturday.

2. Prove that Lake Champlain can be inscribed in a square. More precisely, show that, given any closed curve in the plane, there is a square containing the curve all four sides of which touch the curve. A corner counts for both incident sides.

3. Be the first person ever to prove (or disprove) that every closed curve in the plane contains the corners of some square.

All readers are encouraged to submit prospective puzzles for future columns to puzzled@cacm.acm.org.

Peter Winkler (puzzled@cacm.acm.org) is Professor of Mathematics and of Computer Science and Albert Bradley Third Century Professor in the Sciences at Dartmouth College, Hanover, NH.



**CONNECT WITH OUR
COMMUNITY OF EXPERTS.**

www.reviews.com

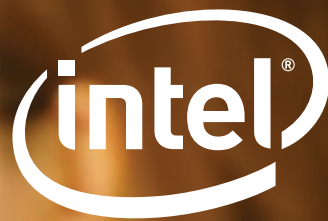


Association for
Computing Machinery

Reviews.com

They'll help you find the best new books
and articles in computing.

Computing Reviews is a collaboration between the ACM and Reviews.com.



Think Parallel....

It's not just what we make.
It's what we make possible.

Advancing Technology Curriculum
Driving Software Evolution
Fostering Tomorrow's Innovators

Learn more at: www.intel.com/thinkparallel

