ACM SIGACT News Distributed Computing Column 28

Idit Keidar
Dept. of Electrical Engineering, Technion
Haifa, 32000, Israel
idish@ee.technion.ac.il

Sergio Rajsbaum, who edited this column for seven years and established it as a relevant and popular venue, is stepping down. This issue is my first step in the big shoes he vacated. I would like to take this opportunity to thank Sergio for providing us with seven years' worth of interesting columns. In producing these columns, Sergio has enjoyed the support of the community at-large and obtained material from many authors, who greatly contributed to the column's success. I hope to enjoy a similar level of support; I warmly welcome your feedback and suggestions for material to include in this column!

The main two conferences in the area of principles of distributed computing, PODC and DISC, took place this summer. This issue is centered around these conferences, and more broadly, distributed computing research as reflected therein, ranging from reviews of this year's instantiations, through influential papers in past instantiations, to examining PODC's place within the realm of computer science.

I begin with a short review of PODC'07, and highlight some "hot" trends that have taken root in PODC, as reflected in this year's program. Some of the forthcoming columns will be dedicated to these up-and-coming research topics. This is followed by a review of this year's DISC, by Edward (Eddie) Bortnikov. For some perspective on long-running trends in the field, I next include the announcement of this year's Edsger W. Dijkstra Prize in Distributed Computing, (presented at DISC'07), along with the list of past laureates. The prize is awarded annually to an outstanding paper on the principles of distributed computing, whose significance and impact on the theory and/or practice of distributed computing has withstood the text of time and has been evident for at least a decade. This year's winner is "Consensus in the Presence of Partial Synchrony" by Dwork, Lynch, and Stockmeyer.

The main part of this issue, contributed by Michael Kuhn and Roger Wattenhofer, is a quest for the "center" of computer science, based on "distances" between conferences. It further studies the evolution of such distances over time, focusing on theory of computing and distributed computing conferences. Finally, it identifies "central" people in these communities. The article extends a more PODC-centered (and quite entertaining) presentation given by Roger at the PODC business meeting.

Many thanks to Eddie, Michael, and Roger for their contributions.

Call for contributions: Please send me suggestions for material to include in this column, including news, communications, and authors willing to write a guest column or to review an event related to distributed computing. In particular, I welcome tutorials, either exposing the community to new and interesting topics, or providing new insight on well-studied topics by organizing them in new ways. I also welcome open problems with a short description of context and motivation.

PODC'07 and Recent Trends

PODC'07, or in its full name, the $26th$ Annual ACM Symposium on *Principles of Distributed Computing*, was held in Portland, Oregon, in August. The acceptance rate this year was more competitive than in the past – 32 regular papers were selected among 204 submissions (roughly 16%). To provide a stage for plenty of more interesting research, the program also included 46 brief announcements, selected among 88 submissions.

Two keynote lectures illustrated how PODC principles are put to practice in the real world. The first, by Marc Tremblay of Sun Microsystems, focused on concurrency in modern microprocessor systems (see more details below), whereas the second, presented by Tom Leighton, discussed Akamai's approach to achieving performance and reliability on the Internet. Three workshops were collocated with PODC this year:

1. *TRANSACT 2007*: The Second ACM SIGPLAN Workshop on Transactional Computing;
2. *LOCALITY 2007*: The Second Workshop on Locality Preserving Distributed Computing Methods;
3. *Dial M-POMC 2007*: The Joint Workshop on Foundations of Mobile Computing.

All three workshops focused on themes that are studied outside the PODC community, as well as within it. They are all related to traditional PODC topics: transactional memory falls squarely within shared memory-based computing; locality is studied mostly in the context of graph algorithms and network algorithms; and mobility draws on a host of topics, also including graph and network algorithms. Yet, there are now new "consumers" for research on these topics: multiprocessors, peer-to-peer systems, and wireless networks. These consumers spark a revival of some traditional PODC topics, causing researchers to look at them in new ways, and also evoke entirely new directions.

Peer-to-peer issues began gaining attention in PODC, as well as in the rest of the computer science world, since 2002. Over the past five years, the PODC program has constantly featured papers about peer-to-peer systems and related graph algorithms, including overlay constructions, compact routing schemes, and peer-to-peer applications. Locality and dynamic algorithms are clearly important in such large-scale systems, and are represented in PODC accordingly.

Two additional topics that have been on the rise in PODC in recent years are (1) distributed storage, boosted by the interest in SAN/NAS technologies; and (2) economic aspects and applications of game theory in distributed systems and networks, which made their debut in PODC in 2004. Future issues of this column are going to address these topics.

Transactional memory, although proposed by Herlihy and Moss as early as 1993, is the "newest kid on the block", showing up in the PODC landscape first in 2005, again, jointly with its steep rise in other communities. I elaborate on this topic and the reasons for its current surge below; stay tuned for a more detailed treatment of the topic in a one of the next issues.

Transactional memory

The PODC'07 program opened with a keynote lecture by Dr. Marc Tremblay, who is a Sun Fellow, Senior Vice President, and the Chief Architect of the Systems Group at Sun Microsystems. Tremblay, who sets future directions for Sun's architectures, discussed the notion of *transactional memory*, and its place in modern microprocessors. In recent years, the literature is abundant with proposals for supporting transactions in programming languages, systems software, and hardware. Tremblay's talk illustrates that this idea is not purely academic. In fact, Sun has recently officially announced that it will include support for transactional memory with the first generation of its Rock processors, which are expected to be out in the second half of next year.

The elevated interest in transactional memory stems from a major paradigm shift undergone by chip manufacturers in recent years– from uniprocessors to chip multiprocessors (CMPs). Current-day chips already include 2, 4, or even 8 cores, and are expected to soon feature many more. This paradigm shift in the hardware world necessitates new software paradigms, which will support highly scalable multi-threaded programming in order to exploit the many cores. It is expected that such software paradigms will be based on shared memory (rather than message passing), given the physical proximity of all cores to common caches and memories. Traditional lock-based thread synchronization does not provide the required level of parallelism, and is complex to program. In its place, transactional memory is emerging as a leading solution for supporting highly scalable multi-threaded programming.

Various aspects of transactional memory are studied in several different research communities, including programming languages, compilers, and computer architectures. The software aspects of supporting and using transactional memory involve multi-process synchronization and shared memory computing models, which have been extensively studied at PODC over many years. It is therefore not surprising that the transactional memory trend has not skipped PODC. A closely related area of research focuses on directly constructing massively parallel lock-free data structures.

The first technical session in PODC this year, immediately ensuing Tremblay's plenary talk, dealt with multiprocessor, multicore, and shared memory issues. This session seamlessly followed the last session of PODC'06, which ended with two papers on lock-free synchronization and contention management for transactional memory. Among PODC's brief announcements, several papers dealt with shared memory synchronization, and in particular, four brief announcements focused directly on transactional memory, as of course did all 11 of the contributions to the TRANSACT workshop.

So what problems in the transactional memory world can PODC help solve? I'll give just one example from this year's PODC below. Expect more on this in one of the upcoming SIGACT News Distributed Computing columns.

One example is presented was presented by a group of researchers from Sun Labs' Scalable Synchronization Research Group. This group has previously developed HyTM, Hybrid Transactional Memory[1], where some transactions are handled directly in hardware, while others, e.g., very large ones, are handled in software. The main challenge in HyTM is to ensure that hardware transactions detect conflicts with software transactions. If software transactions are rare, as one may hope will be the case in future transactional memory systems, then it is advantageous for hardware transactions to check whether there are any on-going software transactions when they begin, and if there are none, perform no further checks. Checking for software transactions can be done, e.g., by reading a global counter. If a new software transaction arrives while the hardware transaction is still on-going, its update of the counter will cause the hardware transaction to abort. However, a global counter solution does not scale well in situations where software transactions

[1] Damron, Fedorova, Lev, Luchangco, Moir, and Nussbaum. "Hybrid transactional memory". In ASPLOS 2006.

are common, and furthermore, it increases the likelihood of aborting hardware transactions due to (false) conflicts on the global counter. One of the papers at PODC this year was directly motivated by the need to improve this performance bottleneck of HyTM: Yossi Lev presented *"SNZI: Scalable NonZero Indicators"*, which he jointly authored with Faith Ellen, Victor Luchangco, and Mark Moir. SNZI is a highly scalable concurrent data structure implementing a presence indicator. Yossi likened SNZI to a light bulb that is on whenever a person is in the room, and off when nobody is in it. The paper shows that, unlike a counter, SNZI can be implemented in a scalable matter, using PODC techniques. While SNZI has many applications, the authors designed it specifically in order to improve the performance and scalability of HyTM, by replacing the global counter with a more scalable SNZI object. The authors have illustrated their algorithm's good performance on transactional memory workloads.

Review of DISC'07

Edward Bortnikov
Dept. of Electrical Engineering, Technion
Haifa, 32000, Israel
ebortnik@tx.technion.ac.il

The $21st$ *International Symposium on Distributed Computing*, DISC 2007, was held in Lemesos, Cyprus, between Sep 24 and Sep 26 2007. The Program committee, chaired by Andrzej Pelc, accepted 32 of 100 regular paper submissions and 9 brief announcements in different areas of distributed computing. The conference featured the presentation of 2007 Edsger W. Dijkstra prize for the seminal paper "Consensus in the Presence of Partial Synchrony" by C. Dwork, N. Lynch, and L. Stockmeyer, three invited talks, and many high-quality paper presentations. DISC 2007 was accompanied by two workshops: COST Action 295 DYNAMO (Algorithms in Dynamic Networks), and IMAGINE - a workshop managed by young researchers (PhD students and fresh graduates). The organizing committee, chaired by Chryssis Georgiou, was very successful in running the conference and the accompanying events at a very high standard.

The lecture hall of the conference venue at the Louis Apollonia hotel was always full, and the discussions during the coffee breaks were vivid, despite numerous temptations offered by the marvelous weather and sea. The famous Cypriot hospitality was felt everywhere. Encouraged by festive address of Limassol's Mayor, the conference's participants set out for a half-day tour which included a visit to the ancient city-kingdom of Kourion, (see Figure 1), and an enchanting dinner in Oleastro (the Olive Park), full of good food and local dances. The event's peak was undoubtedly a Greek dance jointly performed by the Program and Steering Committee Chairs of DISC 2007, Andrzej Pelc and Alex Shvartsman, resp., and those of 2008,

Figure 1: Group picture in the ancient amphitheater of Kourion. Picture courtesy of Chryssis Georgiou.

Gadi Taubenfeld and Rachid Guerraoui, as shown in Figure 2(a). This dance was eventually joined by the community at large, as seen in Figure 2(b).

(a) The handing-over-of-the-chairmanship dance.

(b) The community at-large dancing.

Figure 2: Greek dance, Oleastra (Olive Park). Pictures courtesy of Chryssis Georgiou.

The works presented at DISC covered a large variety of topics - graph algorithms (Shmuel Zaks expressed a hope to extend this part, which prevailed in the old WDAG days), distributed storage, consensus and fault tolerance, networking, self-stabilization, shared memory, and others. There were also entirely new areas, like, for example, distributed mobile robot navigation. The invited talks shed light on three research areas: David Peleg's lecture reviewed the literature on time-efficient broadcasting algorithms for radio networks; Michel Raynal's talk touched his personal viewpoint on the directions to be taken by the distributed computing community and discussed the new timed register abstraction and its applications; Burkhard Monien presented the recent advances in the distributed game theory, namely, routing and scheduling with incomplete information.

The diversity of keynote topics illustrated the broadening scope of problems being addressed by the community. For example, the technical part of Michel Raynal's presentation, based on join work with Gadi

Taubenfeld at SPAA'07, introduced a novel perspective on building shared-memory algorithms for well-studied problems like mutual exclusion, adaptive renaming, consensus, etc. The proposed abstraction, called *timed register*, or TR, is designed for time-based distributed algorithms that exploit synchrony assumptions to provide properties that could be impossible to guarantee in fully asynchronous systems. The algorithms using TRs rely on the existence of a bound Δ such that any pair of consecutive constrained read and write operations issued by the same process on the same TR are separated by at most Δ time units. However, these algorithms cannot afford incorrect behavior when the timing assumptions do not hold. The authors demonstrated how TRs can be used for building *indulgent* algorithms, which can withstand arbitrary periods during which the timing assumptions are violated. During these periods, the algorithms' safety properties are not broken, but progress cannot be guaranteed until the synchrony assumptions are satisfied again. All the presented indulgent algorithms are remarkably simple and elegant. The speaker urged the community to design short algorithms with long proofs, not the vice versa!

Below, I review four representative regular papers. The best student paper award went to David Aisenstat for his work with Dana Angluin and James Aspnes, presenting a population protocol for fast and robust distributed majority computation. The protocol, which is based on random interactions between distributed processes, is inspired by studying random processes within large sets of physical particles, and its analysis involves statistical physics tools. Given an initial configuration, where each of n processes holds one of 3 values (x, y, and $\perp$), their algorithm guarantees that w.h.p. the n processes reach consensus on the majority of x and y within $O(nlogn)$ random interactions, while tolerating $o(\sqrt{n})$ Byzantine processes. The authors further demonstrate the applicability of their results to fault-tolerant register machine construction.

Shlomi Dolev, Seth Gilbert, Rachid Guerraoui, and Calvin Newport studied the problem of gossiping in a multi-channel radio network with a malicious adversary. Their model assumes a single-hop network with multiple non-interfering channels, a set of processes each of which can access one channel in a round, and an adversary that can disrupt one channel in a round. The ε-gossip problem is spreading $(1-\varepsilon)n$ rumors among n parties (note that perfect gossip is impossible since the adversary can block a certain node forever). The authors propose a deterministic gossip protocol (which can be extended to tolerate Byzantine faults), and match its running time with a tight lower bound. A remarkable feature in the lower bound's proof is the usage of the seminal Turan's theorem from the extremal graph theory to reason about the adversary's optimal strategies.

Gadi Taubenfeld presented a paper on efficient transformation of obstruction-free shared memory algorithms into non-blocking algorithms (and published very nice PowerPoint slides online at `http://www.faculty.idc.ac.il/gadi/MyPapers/2007T-obstructionFree.ppt`). The motivation for this work is exploring the spectrum of tools between obstruction-freedom (which is simple but makes weak guarantees) and wait-freedom (which is complex and inefficient but provides strong guarantees). Gadi defined the time complexity measure for analyzing transformations between the models, and showed that the transformation between the two extremes by Fich, Luchangco, Moir and Shavit (DISC 2005) requires exponential time. The paper describes a neat constant-time transformation of an obstruction-free algorithm to a non-blocking one, which employs two copies of the former - one before the critical section, and the other inside it (the details of coordinating between them are not so simple). An interesting observation is that obstruction-freedom is useful even when locks are employed. The question of existence of a time-efficient transformation between non-blocking and wait-free algorithms remains open.

Baruch Awerbuch and Christian Scheideler considered the problem of building denial-of-service (DOS)-resilient distributed hash tables (DHT's). They considered adversaries that know everything about the system until some time, and can block some servers after this time. The proposed strategy employs $O(log^2 n)$ replicas per data item, using multiple hash functions. The lookup protocol is a nice, albeit complex, combination

of the contraction stage, in which requests are forwarded towards the hashed data items via multiple random routes, and the expansion stage, in which the requests stalled at the blocked nodes explore exponentially increasing network neighborhoods in order to search for the stored replicas. The authors prove that a sequence of requests can be handled with polylogarithmic overhead, assuming that the maximal possible fraction of blocked servers is $1/144$ (leaving a lot of room for improving constants).

Dijkstra Prize 2007

The 2007 Edsger W. Dijkstra Prize in Distributed Computing was awarded, in DISC'07, to the paper "Consensus in the presence of partial synchrony" by Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer, which appeared in the Journal of the ACM, Vol. 35, No. 2, April 1988, pages 288–323. I include here a reprint of the award statement.

This paper introduces a number of practically motivated partial synchrony models that lie between the completely synchronous and the completely asynchronous models, and in which consensus is solvable. It gave practitioners the right tool for building fault tolerant systems, and contributed to the understanding that safety can be maintained at all times, despite the impossibility of consensus and progress is facilitated during periods of stability. These are the pillars on which every fault tolerant system has been built for two decades. This includes academic projects such as Petal, Frangipani, and Boxwood, as well as real life data centers, such as the Google file system.

In distributed systems, balancing the pragmatics of building software that works against the need for rigor is particularly difficult because of impossibility results such as the FLP theorem. The publication by Dwork, Lynch, and Stockmeyer was in many respects the first to suggest a path through this thicket, and has been enormously influential. It presents consensus algorithms for a number of partial synchrony models with different timing requirements and failure assumptions: crash, authenticated Byzantine, and Byzantine failures. It also proves tight lower bounds on the resilience of such algorithms.

The eventual synchrony approach introduced in this paper is used to model algorithms that provide safety at all times, even in completely asynchronous runs, and guarantee liveness once the system stabilizes. This has since been established as the leading approach for circumventing the FLP impossibility result and solving asynchronous consensus, atomic broadcast, and state-machine replication.

In particular, the distributed systems engineering community has been increasingly drawn towards systems architectures that reflect the basic split between safety and liveness cited above. Dwork, Lynch, and Stockmeyer thus planted the seed for a profound rethinking of the ways that we should build, and reason about, this class of systems. Following this direction are many foundational solutions. First, these include state machine replication methods such as Lamport's seminal Paxos algorithm and many group communication methods. Another important branch of research that directly follows this work is given by Chandra and Toueg's unreliable failure detector abstraction, which is realized in the eventual synchrony model of this paper. As Chandra and Toueg write:"we argue that partial synchrony assumptions can be encapsulated in the unreliability of failure detectors. For example, in the models of partial synchrony considered in Dwork et al. it is easy to implement a failure detector that satisfies the properties of $\Diamond W$." Finally, the insight by Dwork, Lynch, and Stockmeyer also led to various timed-based models of partial synchrony, such as Cristian and Fetzer's Timed-Asynchronous model and others.

The award committee would like to acknowledge the sincere efforts by the nominators of this work, as well as all other (worthy!) nominations which came short of winning.

The committee wishes to pay a special tribute via this award to Larry Stockmeyer, who passed away on July 31, 2004. Larry's impact on the field through this paper and many others will always be remembered.

The Committee of the 2007 Edsger W. Dijkstra Prize in Distributed Computing

- Hagit Attiya
- Dahlia Malkhi
- Keith Marzullo
- Marios Mavronicolas
- Andrzej Pelc
- Roger Wattenhofer (Chair)

Past Prizes

2006: John M. Mellor-Crummey and Michael L. Scott for "Algorithms for scalable synchronization on shared-memory multiprocessors", ACM Transactions on Computer Systems, 9(1), 1991.

2005: Marshal Pease , Robert Shostak and Leslie Lamport for "Reaching agreement in the presence of faults", Journal of the Association of Computing Machinery, April, 1980, 27(1):228-234.

2004: R. G. Gallager , P. A. Humblet and P. M. Spira for "A Distributed Algorithm for Minimum-Weight Spanning Trees", ACM Transactions on Programming Languages and Systems, January 1983, 5(1):66-77.

2003: Maurice Herlihy for "Wait-Free Synchronization", ACM Transactions on Programming Languages and Systems, January 1991, 13(1):124-149.

2002: Edsger W. Dijkstra for "Self-stabilizing systems in spite of distributed control", Communications of the ACM, 1974, 17(11):643-644.

2001: Michael J. Fischer , Nancy A. Lynch and Michael S. Paterson for "Impossibility of Distributed Consensus with One Faulty Process", Journal of the ACM, April 1985, 32(2):374-382.

2000: Leslie Lamport for "Time, Clocks, and the Ordering of Events in a Distributed System", Communications of the ACM, July 1978, 21(7):558-565.

Prizes in the years 2000–2002 were given under the name "PODC Influential-Paper Award".

The Theoretic Center of Computer Science[2]

Michael Kuhn and Roger Wattenhofer
Computer Engineering and Networks Laboratory
ETH Zurich
8092 Zurich, Switzerland
{kuhnmi,wattenhofer}@tik.ee.ethz.ch

Abstract

In this article we examine computer science in general, and theory and distributed computing in particular. We present a map of the computer science conferences, speculating about the center of computer science. In addition we present some trends and developments. Finally we revisit centrality, wondering about the central actors in computer science. This article is a printed version of a frivolous PODC 2007 business meeting talk, held by the second author. In an attempt to address a broader audience we have extended the content by data about theory conferences and computer science in general. Still, the character of the information remains the same, meaning our investigations should not be taken too seriously.

1 Introduction

People have always been fascinated by centrality. They have been afraid of the middle of the night, and estimated time based on the middle of the day. Most people also like to be the center of attention, and look at things in egocentric ways. Not surprisingly, the Babylonians placed themselves in the middle of their world map. And even today, almost any country claims to be the center of its continent. Noteworthy is surely also the question about the center of the universe. Among many others Ptolemy, Copernicus, and Galileo have thought about it, some of them at the risk of life.[3]

Recently, when presenting some material about conference similarities, we have been asked a question of similar nature by a colleague. He was curious about the center of computer science. In a declaredly navel-gazing attempt to address this question we will outline a map of computer science in the beginning of the article. We will then present different facets of the world of computer science. Being genuinely more risk-averse than Galileo et al.[4] we would like to stress that all our investigations should be taken with a grain of salt.

[2] ©Michael Kuhn and Roger Wattenhofer, 2007

[3] At the time the Catholic Church accepted Galileo's model, physicists have already claimed that there is no center of the universe. Contradicting the physicists, Fremont, WA declares itself the center of the universe. Other eccentric centers of the universe are the Toronto Maple Leaf hockey team, or a manhole cover in Wallace, Idaho.

[4] Not to mention Giordano Bruno!

2 Computer Science Cartography

Maps have always been an essential instrument in exploring space. It was, after all, the Ptolemy world map—and its mistakes—that triggered Columbus' monumental voyage and the discovery of America. Reason enough to draw a map of "our world", the world of computer science.

2.1 Method

We first set up a model for the world of computer science. Assuming that computer science conferences adequately represent computer science disciplines, we model the interrelations between the conferences as a graph.

To construct such a graph, we introduce a notion of "social conference similarity". Basically, two conferences are supposed to be closely related if they exhibit a large number of common authors. To avoid overestimating the similarity of massive events—they naturally have a large number of common authors—we improve on this idea by incorporating a normalization method: Consider two conferences, C_1 and C_2, that contain a total of s_1 and s_2 publications, respectively. Further, assume that there are k authors A_i ($i = 1, ..., k$) that have published in both of them and that author A_i has $p_{i,1}$ publications in conference C_1 and $p_{i,2}$ publications in conference C_2. We can now define the similarity $S(C_1, C_2)$ between C_1 and C_2 as follows[5]:

$$S(C_1, C_2) = \sum_{i=1}^{k} \min\left(\frac{p_{i,1}}{s_1}, \frac{p_{i,2}}{s_2}\right)$$

Applying this similarity measure to all pairs of conferences results in the desired graph, our model of the world of computer science. The required information for this graph was extracted from the DBLP bibliographic repository.

Luckily, the "cartography of graphs", better known as graph embedding, is a well explored topic. Simply speaking, the goal of a graph embedding algorithm is to assign Euclidean coordinates to all vertices of a graph, such that the resulting positions well reconstruct the graph distances between all pairs of nodes (also multi-hop).[6] Similarly as it is impossible to undistortedly draw the globe in 2 dimensions, it is in general also not possible to exactly represent a graph metric in 2 dimensions.

Out of the rich choice of embedding algorithms we have opted to apply the widely used multi-dimensional scaling method (MDS) to create our map. MDS minimizes the mean absolute squared error when comparing all the pairwise distances in the graph metric (i.e. shortest paths) and the resulting Euclidean metric.

It is important to note that there is no notion of orientation for such an embedding. It only represents pairwise distances, and would be equally valid after applying any congruence transformation.

2.2 Results

Figure 3 shows the map of computer science conferences, constructed as described in the previous section. To keep the map readable we have restricted the vertex set to only include top-tier conferences. The required

[5]There are surely a couple more normalization methods one could think of. Manual inspection of the options we tried, revealed this to be the most suited.

[6]More generally, graph embedding is the process of mapping a graph into any other space (not necessarily Euclidean), thereby pursuing some (not necessarily distance related) design goals.

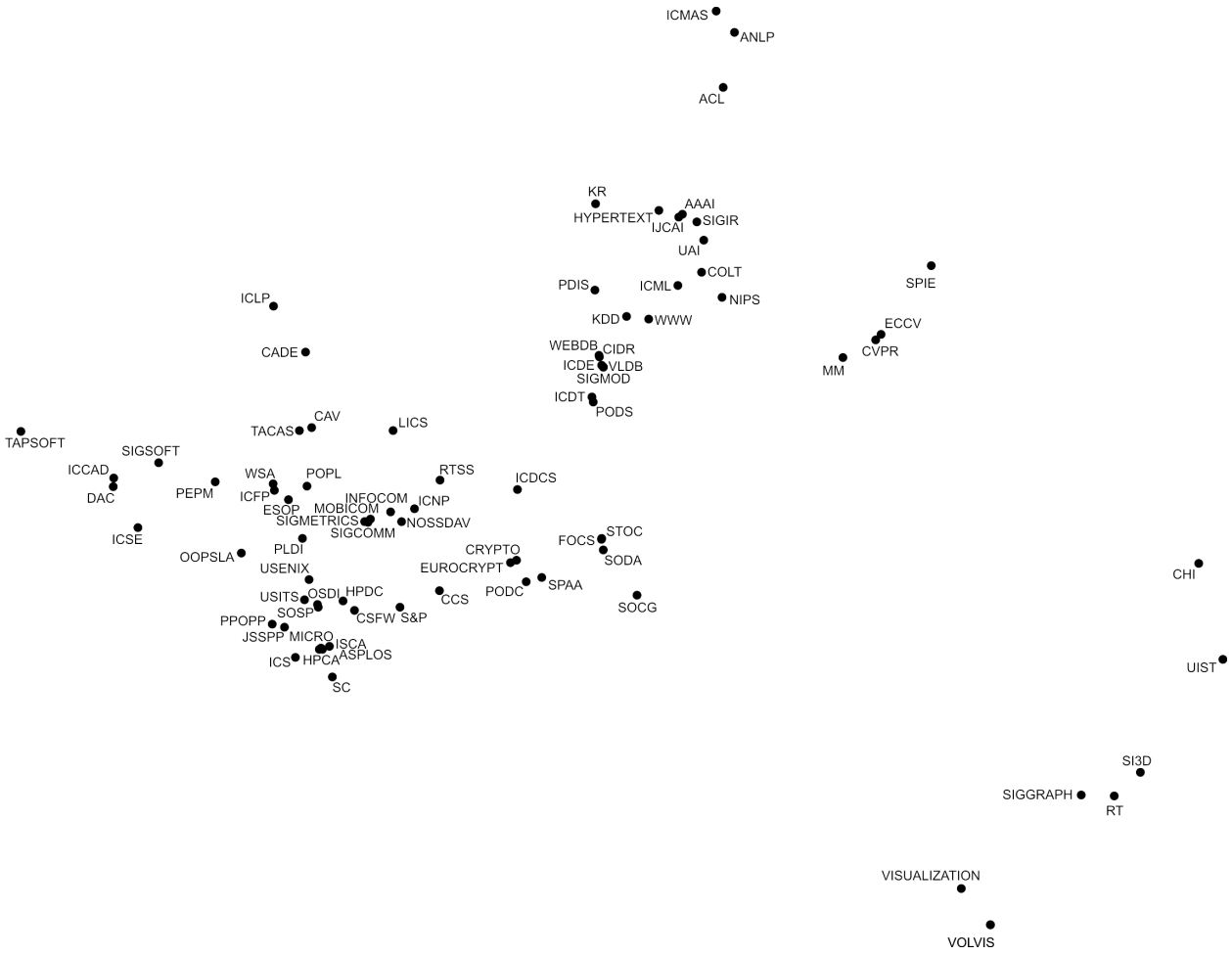


Figure 3: Our map of computer science: The map was constructed by embedding the conference graph into a 2-dimensional Euclidean space. Only top-tier conferences (according to Libra) are shown. Note that the map only represents pairwise distances, there is no notion of orientation, i.e. the axes can be chosen arbitrarily.

conference classification—with which the authors disagree to a certain extent—was taken from *Libra*[7].

Our map gives room for quips and questions. It can, first of all, be observed that like-minded conferences tend to gather together, they build families and tribes. It is thus possible to investigate the interconnections between entire "scientific cultures" rather than single conferences. Software engineers and design automation specialists, for example, live in the west of this world.[8] Their eastern neighbors are mainly into computer networks, and further south hardware architecture and operating systems experts are located.

In the very north, people mainly study natural languages and artificial intelligence. Also, they retrieve information from and mine the data provided by their direct southern neighbors (databases and the world wide web). The south-eastern population, finally, lives quite independently, and is mostly interested in user interfaces, graphics and visualization. It is interesting that these disciplines almost "drop" from the disk, and that we experience these large distances in the south-eastern part of our map. Are these gaps merely an artifact of our sloppy model, making our drawing as inaccurate as the disk-shaped Babylonian world map, or does this emptiness call for attention by the computer science community?!

We observe that the center of the graph is dominated by theory, and—slightly less significantly—also cryptography, distributed computing, networks, and databases. Interestingly, all these areas can be seen as service suppliers for other disciplines: Data can hardly be mined without databases, software design requires a great amount of (algorithmic) theory, and networks as well as cryptography and distributed systems play a crucial role in many real world systems. Hence, looking at our map, a theoretician could, with a healthy dose of self-esteem, derive that he (or at least his field of research) is in the center of computer science.[9]

Our theoretician would surely like to get a more detailed view on his self-declared center of computer science. Even though we question the center predicate, we do him—and hopefully also our theory-focused readers—a favor and zoom into this section. We have inserted the lower-tier conferences (again, according to Libra, and again, we do not fully agree) into the drawing by placing them into the weighted center of their closest top-tier neighbors (while keeping the layout of these top-tier conferences fixed). Figure 4 illustrates the theory close-up of our map.

In an attempt to interpret the figure, one might notice a slight separation of west and east. The west-side of the map seems to be mostly populated by the species of "practical theoreticians", while the east-side is rather inhabited by "pure theory". Interestingly, the cryptographers squeeze themselves into the territory of distributed computing. Clearly the two share common roots, but still the proximity is remarkable. Comparing the distances in the map to the distances in the original graph reveals that there is quite some discrepancy in this specific case. We speculate that distributed computing and cryptography get intermixed because they both lie between (pure) theory and systems/networking. Possibly, a 2-dimensional map can not accurately represent the ultimate truth. As detailed views usually provide deeper insights into a problem, the insight of this close-up is perhaps that our map should be treated with care.

3 Migration in Computer Science

As the history of mankind, the world of computer science is not static. Communities emerge, disappear, and migrate. In the following we want to have an eye on the movements in the proximity of PODC, as well as

[7] `http://libra.msra.cn/` For each discipline the site lists the major conferences grouped by tier (e.g. `http://libra.msra.cn/conf_category_1.htm`)

[8] Again, note that there are no axes defined, we just orient ourselves as the picture is aligned here.

[9] Clearly, the map leaves room for discussion. A closer look reveals many surprises, e.g. the location of design automation conferences. Also, other (equally reasonable) centers, such as databases or networking could doubtlessly be identified—please mind that we write this article for SIGACT and not SIGMOD or SIGCOMM. More fundamentally one might wonder whether there at all is a center of computer science; maybe we rather live in a "centerless" world, much like modern physics sees the universe.

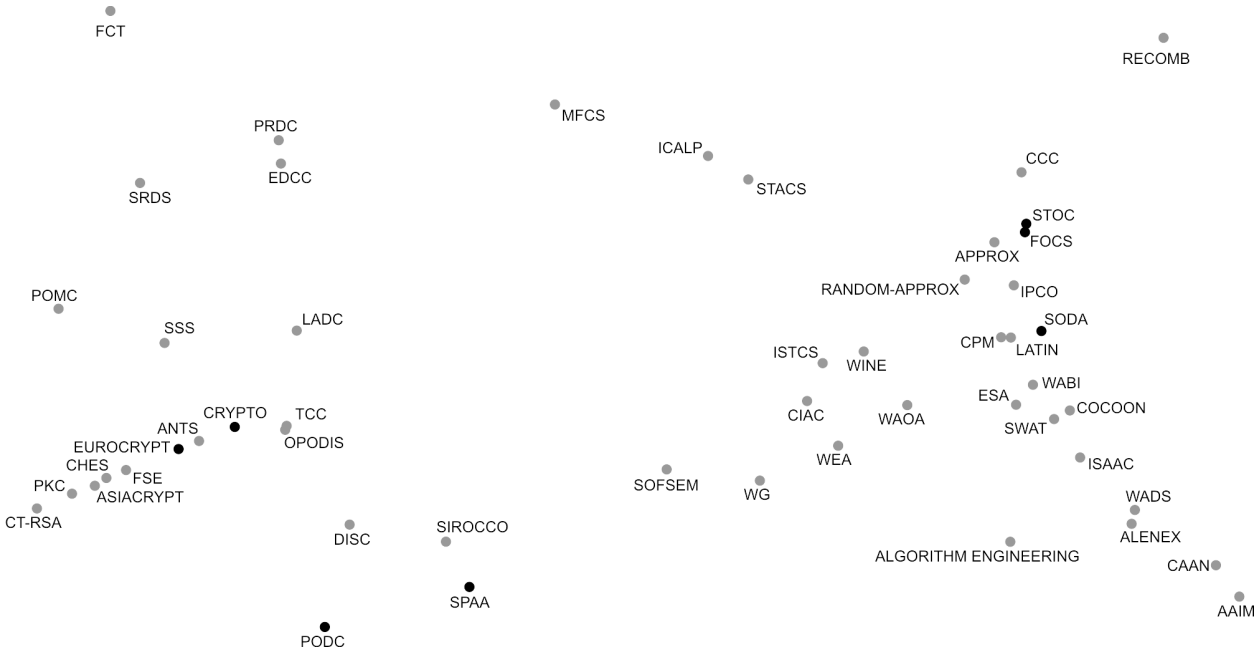


Figure 4: Close-up of the map around the theory conferences. Tier-1 conferences (according to Libra) are marked black, any other conferences gray.

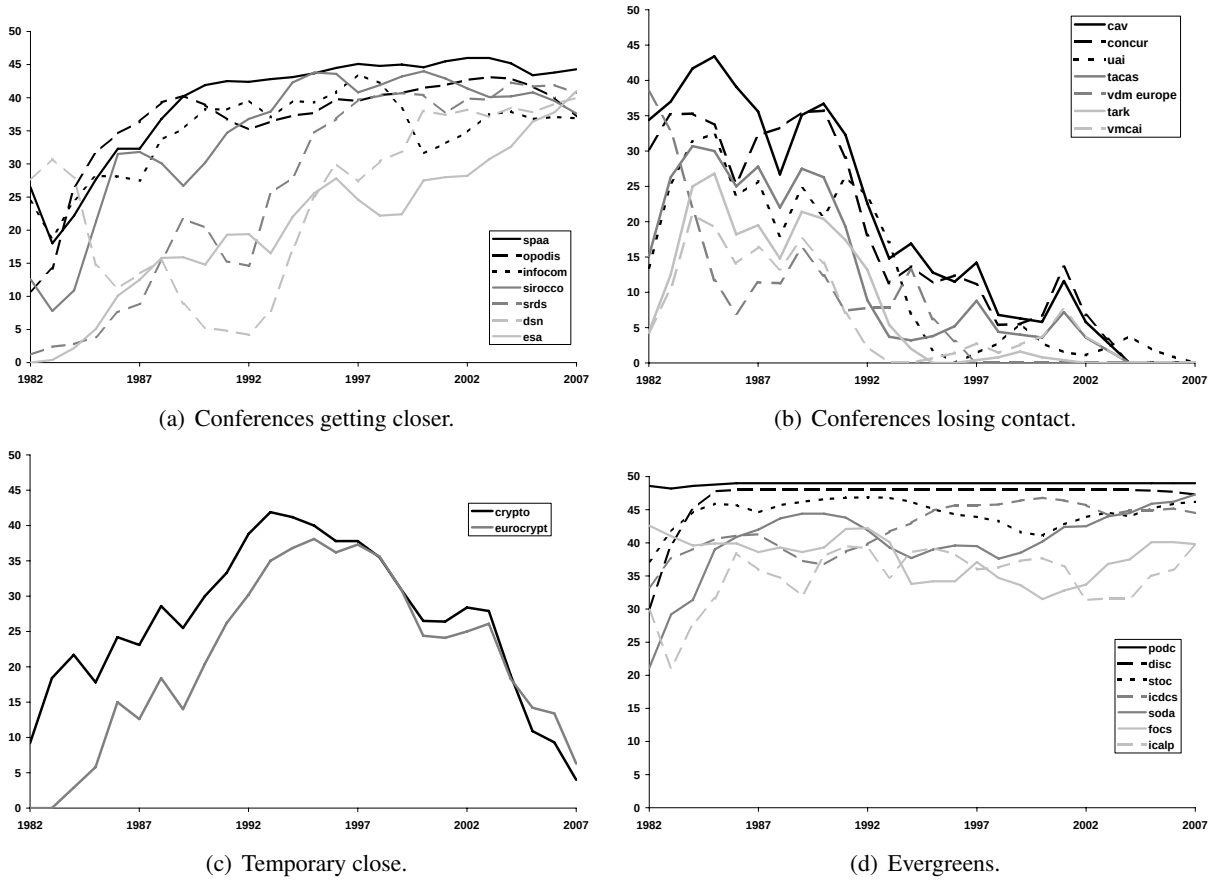
some major theory conferences, namely STOC, FOCS and SODA (which we in the following will treat as one).

3.1 Method

A conference is mainly defined by its participating authors. We thus assume that looking at the changes in authorship is a handy method to capture the changes of a conference over time. Consequently, we have applied the idea of our "social similarity measure" from Section 2.1 on a per year basis: For a particular year and conference we have examined where else the authors would typically publish (at all times). This gives, for this particular conference, an insight in what other conferences have been particularly close at a given point in time.

3.2 Results

The described "time dependent social similarity measure" allows to plot the development of a conference's proximity over time. Figures 5 and 6 show such plots for PODC and STOC/FOCS/SODA. An interesting observation is, maybe, the temporal closeness of cryptography to both, PODC and the theory conferences. Most likely, this does not mean that cryptography is in danger of extinction, but rather protocols an emancipation process; cryptographers have grown strong enough to form their own, independent community and thus drift away from other conferences.



(a) Conferences getting closer.

(b) Conferences losing contact.

(c) Temporary close.

(d) Evergreens.

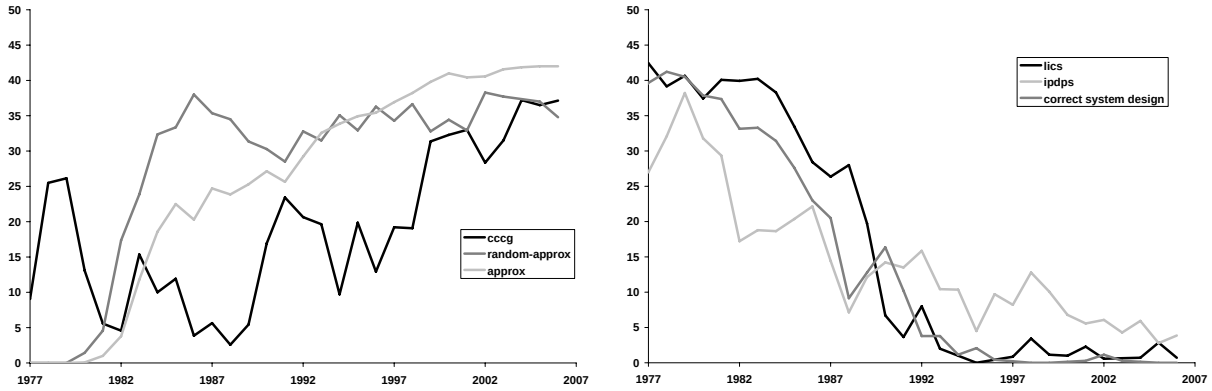
Figure 5: Related conference trends for PODC.

4 Keyword Trends: Predicting the Perfect Paper Title

After this brief excursion into the evolution of conference relationships, we want to come back to a more global view. How did computer science evolve over time? We believe that terminology is an meaningful witness for scientific history. We thus analyze the change of central keywords in computer science. Moreover, we will, in a tongue-in-cheek way, suggest the perfect titles for the next year's edition of PODC as well as STOC/FOCS/SODA.

4.1 Method

As usual when it comes to predicting future trends we rely on historic data. For this purpose, history has been divided into 6 time-slots, from 1977 to 2006 in 5-years chunks. Moreover, for each conference in question we have parsed all the titles appearing in its proceedings to extract the single words. Counting the number of occurrences for each word and time-slot allows to establish a "per-time-slot" ranking which can then be mined to extract trends. We have restricted the search space to keywords that made it into the top-50 in at least one time-slot. The actual trend analysis (i.e. selection of the keywords that best characterize the movements) was then mainly carried out manually.



(a) Conferences getting closer.

(b) Conferences losing contact.

(c) Temporary close.

(d) Evergreens.

Figure 6: Related conference trends for STOC/FOCS/SODA.

4.2 Results

How did computer science research evolve over time and where does it go? Our analysis shows that *wireless mobile agents based on neural learning that quickly adapt to image and video web services* are fashionable.[10] On the other hand, we are no longer interested in *computer experts specifying the semantics of parallel relational databases for VLSI in the prolog programming language*.[11] Also, *knowledge about object oriented simulation logic* does not seem to be required any longer, even though the topic was a hot in the beginning of the 90ties.

If you do not want to risk being labeled antediluvian after submitting to a major theory conference, better name your paper, say:

> *"Online Quantumn Algorithms to Approximate Directed Location Codes"* or *"On the Hardness of Scheduling using Random Sampling"*

Moreover, avoid titles like

[10] In other words, the keywords web, agent, wireless, neural, mobile, video, learning, image, adaptive, services are becoming increasingly popular.

[11] Again, we witness that the keywords computer, expert, etc. have seen their peak.

"Relational Logic to Separate Automata Isomorphism Classes" and *"Probabilistic Parallel Programs for VLSI Design"*

as these will immediately out you as being stuck in the 80ties. Similarly, the next year's blockbuster of PODC is rather going to be called

"Failure Detectors and Scalable Dynamic Quorum: A Free Mobile Ad-Hoc Game with Selfish Peers"

than, for example

"Recover from Committed Ring-Deadlocks using Message Passing Communication, Temporal Knowledge and Parallel Computation Processes"

5 Kevin Bacon of Computer Science

After temporarily drifting away from the centrality question, we want to come back to it again in this section. Maybe it was short-sighted to ask *what* the center of computer science is. Why not asking about the central actor of computer science, i.e. *who* the center of computer science is? Science is, after all, made by people, and not disciplines.

5.1 Method

The central actor? Well, it is widely known that this is Kevin Bacon—in the movie industry.[12] In math, the corresponding role is accredited to Paul Erdős. In a similar approach we attempt to nominate the central actor in computer science as well as STOC/FOCS/SODA and PODC. Analogously to the construction of the Erdős number, we base our method on the co-author-graph. We then create the induced subgraph for each region of interest (PODC, STOC/FOCS/SODA, and computer science). For PODC, for example, this graph would only contain authors that have at least one PODC paper, and so on.

Other than in the construction of the Erdős number, we do not rely on shortest paths, but rather on the PageRank idea: We start several short random walks at different nodes of the graph, and count how often each author gets visited.[13] This idea is then extended to time dependent centrality, by starting the random walks only at authors that have published in the last five years.

5.2 Results

Table 1 lists the central actor of computer science, as well as his major competitors. Oddly enough, our central actor, Alberto L. Sangiovanni-Vincentelli, acts quite at the edge of our computer science map.[14] But which is right, the map or the election? Critics might ask us to rethink our election process—as oppositions always do after elections. Maybe they are right. Sangiovanni-Vincentelli's toughest competitors, though, seem to live much more in the heartland of computer science—mostly in the area of databases. Maybe there is nonetheless a grain of truth in both, the map and the election?

[12] This is not perfectly true. Rod Steiger is said to be the best connected actor (source: Wikipedia article on the Bacon Number).

[13] More precisely, the number of walks is proportional to each author's number of Tier-1 papers.

[14] He has most frequently published in the area of design automation (DAG, ICCAD, DATE) and his research also covers electrical engineering. Note that both, the map and the author election base on the same set of publications, such that we cannot assume that electrical engineers get preferred by the election process.

All-time	Last 5 years
Alberto L. Sangiovanni-Vincentelli	Wei-Ying Ma
Noga Alon	Wei Wang
Hector Garcia-Molina	Hector Garcia-Molina
Michael Stonebraker	HongJiang Zhang
Michael J. Carey	Noga Alon
Yishay Mansour	Philip S. Yu
Moshe Y. Vardi	Christos Faloutsos
Christos Faloutsos	Gerhard Weikum
David Maier	Joseph M. Hellerstein
Philip S. Yu	Jiawei Han

Table 1: The most "central" authors in computer science. Note that name ambiguities might have distorted the results.

All-time	Last 5 years
Noga Alon	Noga Alon
Avi Wigderson	Avi Wigderson
Robert Endre Tarjan	Sanjeev Arora
Frank Thomson Leighton	Moses Charikar
Moni Naor	Anupam Gupta
Yishay Mansour	Madhu Sudan
Oded Goldreich	Erik D. Demaine
Richard M. Karp	Alan M. Frieze
Amos Fiat	Uriel Feige
Michael E. Saks	Yishay Mansour

Table 2: The most "central" authors in STOC/FOCS/SODA.

The critics would probably react even more sharply when looking at our central actors over the past 5 years. Indeed, there are some surprises in this list. Maybe this is a stunning proof of the often-quoted Asian research momentum, maybe it is caused by name ambiguities that even get reinforced by PageRank-like algorithms[15] (and most likely, it is a combination of both).

Clearly, the nomination of a single representative for the entire world of computer science is a delicate task, and the result inevitably controversial. Imagine, you had to elect a single person representing our planet in front of the universe—another unenviable task. In the following we thus focus on restricted areas of research. Elections within a single culture seem more realistic. Not surprisingly, Noga Alon, second in the list of computer science, is attributed the honor of the central player in theory (see Table 2).

For the PODC community, Nancy Lynch is found on top, as shown in Table 3. All in all, there seem to be less surprises in top-10 lists of the theory cluster and PODC. Most likely, name ambiguities play a less important role in these more restricted areas, as they contain a lower number of people. If you could not find your-

[15]Persons holding an ambiguous name do not only get more weight themselves, but also pass this weight to their collaborators, therefore the reinforcement. We want to stress that the (top-ranked) name Wei-Ying Ma does not seem to suffer from such ambiguities (as opposed to others). Wei-Ying Ma might or might not have profited from "passed on" ambiguities just as anybody else in the list.

All-time	Last 5 years
Nancy A. Lynch	Rachid Guerraoui
Danny Dolev	Nancy A. Lynch
Hagit Attiya	Roger Wattenhofer
Yehuda Afek	Danny Dolev
Maurice Herlihy	C. Pandu Rangan
Nir Shavit	Hagit Attiya
Rachid Guerraoui	Michel Raynal
Sam Toueg	Maurice Herlihy
Michel Raynal	Idit Keidar
Yishay Mansour	Sam Toueg

Table 3: The most "central" authors in PODC.

self on the lists, feel free to check the extended versions on `http://www.confsearch.org/ca.jsp`.

6 Conclusion

What is the center of computer science? A controversial question. Some might claim that computer science is all about building computing machines—possibly only theoretically. Others think it is all about the art of programming these machines, or about describing languages to talk to them. Yet other people associate computer science with algorithms, and believe P vs NP is at the heart of computer science. Finally, the tremendous impact of the Internet, or the huge data collections in service today might speak in favor of networking or database experts, respectively. In this article, we have tried to highlight this controversial nature by examining various aspects from different points of view. We have claimed persons to be in the center of computer science, and at the same time drawn a map placing them at the very border. We have looked into the evolution of scientific disciplines and observed that the same evidence can be interpreted as both, the birth as well as the funeral of a research area. Without doubt the future will teach our evaluations a lesson, ultimately revealing in which direction computer science evolves, and maybe even discover the most influential computer scientist. After all, research is not about how many papers we write, or how many citations they get, but rather, what the best contributions are.