Efficient Algorithms and Data Structures

Mikkel Thorup

Abstract

The proposed project will address some of the fundamental issues in efficient algorithms and data structures, ranging from pseudo-random hashing, to the existence of deterministic dictionaries with constant update and look-up time, to graph algorithms.

The applicant is a leading figure in the area of efficient algorithms and data structures. For the last 14 years he has been in the USA where the area has its stronghold. The grant will facilitate his return to the University of Copenhagen and will enable him to create a mini-center of excellence: a new focal point that will boost the research in efficient algorithms and data structures in Europe.

Project description

Efficient Algorithms and Data Structures (with references to some related work of mine) Algorithms is one of the main areas of computer science, both in terms of teaching and research. It is relevant to the processing of data whenever we need scalable solutions, hence for a lot of science and industry. Suppose, for example, that we want to solve a problem involving n variables. A trivial exhaustive search algorithm might try all combinations. However, even if each variable has only two options, there are 2^n combinations, which is *exponential*, and if $n \geq 70$, then this is too much even if all the worlds computers worked on it for a year. In algorithms we try to find solutions in time *polynomial* if not *linear* in the number of variables. Since any general enough problem is NP-hard or worse, we will have to understand and exploit the special nature of the problem at hand. Take a classic problem like that of sorting n numbers. Sorting is commonly needed in the processing of data. A naive algorithm like Insertion Sort takes one number at the time, scanning for its position among those already sorted, in $O(n^2)$ total time. The 'O' represents an unspecified constant depending on the concrete computer used. Our focus in algorithms is asymptotic performance with $n \to \infty$. The more sophisticated Quick Sort solves the problem in $O(n \log n)$ expected time, allowing us to handle much larger data sets. I myself have the record with an algorithm sorting n integer or floating point numbers in $O(n\sqrt{\log\log n})$ expected time [25]. It is still a major open problem if integers can be sorted in linear time. Generally speaking, in *algorithms* we try to understand how well we can solve combinatorial problems making the most efficient use of computational resources such as time and space. The discipline is theoretical in that we try to prove that our solutions are efficient for all possible inputs, or alternatively, in expectation for a given input distribution.

Many algorithms researchers have their main focus on deciding which kind of problems can be solved in polynomial time (see e.g., [49]). In particular, for NP-hard optimization problems, a lot of effort is put into understanding how good an approximation we can guarantee in polynomial time (my papers [4, 11, 27] follow this line of work). However, as in the above sorting example, I am very interested in more *efficient algorithms* where polynomial time is not enough, but where we try to get the running time closer to linear. The underlying motivation is to understand how to deal with large data sets. Very concretely, it is not uncommon to deal with data sets with, say, $n \geq 10^8$ elements. On today's computer, simple instructions like a memory lookup take about 10^{-7} seconds. A quadratic algorithm using n^2 instructions would take more than 30 years. The issue is not going to be resolved by more powerful computers, it is only going to get worse: the size of the problems considered tends

to grow with the size of the memory which again tends to grows at least as fast as the processor speed. The need for more efficient algorithms is therefore ever growing. Parallel computers might save a factor n, but the principal problem remains: we still need low degree polynomial time. Also, to minimize communication delays, we would still want the individual processor to do as much work locally as possible. This proposal considers hashing which is equally relevant in sequential, parallel, and distributed settings.

An important part of efficient algorithms is data structures where the focus is on how we can efficiently represent information. The planar distance oracle from my selected paper [43] is a good example where I reduce the space needed for fast queries from the trivial $O(n^2)$ to $O(n \log n)$. In dynamic data structures we also want to support changes efficiently, e.g., in priority queues [48], we can insert numbers and extract the minimum. Priority queues are used both directly and inside many greedy algorithms, e.g., to speed up shortest paths in directed graphs. The dynamic graph algorithms from [26] can maintain the bridges in a dynamic graph, and this has lead to efficient implementations of classic inductive proofs in matching theory. One of the most fundamental data structures is a dictionary or hash table which allows us to store and look up information associated with keys. The problem is a bottleneck for many kinds of data analysis including the processing of high volume data streams. It also forms the inner loop of many algorithms. The problem has been central to computing as long as we have had computers (my papers [2, 38, 52, 53] follow this line of work). The study of efficient (not just polynomial) algorithms and data structures has long proud tradition within computer science including the Turing awards of Knuth, Hopcroft, and Tarjan.

Finally note that the theory is not practice, but theory can give a great basis for practice. The Google founders came from a back-ground in theory/mathematics. My student Stephen Alstrup's start-up Octoshape is another good example, and I hope that Stephen will join University of Copenhagen promoting start-ups. I myself got the AT&T Fellow Honor for my technological impact (part of my AT&T job is to consult on the application of theory in concrete practical projects, e.g., speech recognition, Internet traffic routing, management, and analysis, wireless including iPhones etc.).

Aim for technical understanding Algorithms is trying to understand how we can most efficiently solve computational problems. One can distinguish between two types of contributions. We have the problem pioneers that broaden the field identifying new types of problems, and we have the technical pioneers like me who try to deepen the field, developing new powerful techniques to solve the problems. The two sides are in a dynamic interplay. When new problems are identified, the

first task is to understand how they can be addressed with existing techniques. This also provides us with a better understanding of the breadth of these techniques. If existing techniques do not provide a satisfactory solution and if the problem has lasting relevance, then the problem solvers are challenged to develop new powerful techniques that the problem pioneers, in turn, may later apply to more new problems. A good analog is medicine, thinking of problems as diseases and algorithms as treatments. Studying new diseases like the bird flue is important, but so is a new treatment for an old disease like cancer. If your goal is to treat people, then it may be less important exactly what the disease is. In fact, the big hope is to find treatments that work for many types of diseases. Data structures is a great example of this generality since we focus on black-boxes that help in algorithms addressing many different problems.

The frontier of our technical abilities is often best seen in problems of well-known importance but where existing techniques have been exhausted with no satisfactory solution in sight. This is when we know that original technical ideas are needed. To illustrate the type of research I aim for, let me quote a referee report on my selected paper "Compact oracles for reachability and approximate distances in planar digraphs" [43]: *The paper considers the problem of constructing a data structure for static directed planar graphs supporting reachability queries and approximate distance queries. For reachability queries [is there a directed path from one vertex to another] the author achieves* $O(1)$ *time with space* $O(n \log n)$. *Previously no data structure was known with constant time queries and using* $o(n^2)$ *space! The data structure for approximate distances is a generalization of the reachability data structure. The main technical contributions of the paper are contained within the reachability result (the decomposition of a planar graph into a set of 2-layered graphs and the dipath decomposition of these; amazingly simple!). The problems considered have previously been intensively studied by many authors, but the present paper is the first to make essential progress on the problems. I consider the results of the paper to be breakthrough results.* Note that simplicity is a virtue. Generally we prefer proofs to be simple, and it is particularly important that algorithms are simple if we hope for practical impact.

Fundamentally speaking, my focus is more on solutions than on problems. What really excites me are new simple elegant techniques with the power to solve important problems. The last criteria means that my starting point is always an important (still relevant) problem beyond our current capabilities. The problem itself is not the only target. The hope is for surprising and powerful solutions/techniques with potential impact beyond the problem considered. It is this focus on solutions and technical understanding that has lead me to breakthroughs spanning from pure mathematics [33],

to theory [26, 50, 43, 47, 48, 44, 38, 39] to practice [15, 16, 20, 38] (see [19] for context of [20]).

Plans, feasibility, and novelty For my kind of research, novelty and feasibility are not to be found in the initial plans and approaches. This is basic theoretical research looking for novel, surprising, simple, and powerful techniques. The feasibility of this objective is to be found in the general approach and skills, as documented by my track record. The issue is nicely illustrated by the following quote from the expert referee report on my paper "The minimum k-way cut of bounded size is fixed-parameter tractable" with Kawarabayashi [29]: *The techniques used in the algorithm are very ingenious and were never before used for parameterized cut problems. The paper is completely elementary and self-contained (in particular, no tools from graph minor theory is used). In fact, the paper essentially contains no proofs: once they explain the algorithm, it is all obvious! This does not mean that the algorithm is trivial: the ideas are quite unnatural and it is not obvious in the beginning why this approach should work at all, but apparently they do. Having such a "simple" proof should be considered as big plus.* The point here is that the novel successful approach was unnatural to the expert (I was not an expert but had the outsiders fresh inspiration), so even if I had put it in a project plan, then it would not have been feasible-looking. In fact, since the technique is simple, if it had been natural, then it would have been found long ago. Moreover, we tried lots of novel approaches before we got this one to work. More generally, the basic requirements for STOC/FOCS level research is that the problem should be important and the technique surprising. Thus we are looking for surprisingly powerful ideas, but those you cannot plan for. Many ideas are developed, tried, and failed as work with the problem. When first you have the right idea, it often does not take so long to check if it works. Before you have proved that an idea works, you do not know if you are on the right track. Finally, with ambitious targets, there is a very good chance that the problem will turn out too hard, if not be impossible. Staying focused on a single ambitious target is thus very risky.

A feasible flexible general approach My research strategy is not unique but shared by many successful researchers in theory. Striving for excellence, I often aim at ambitious targets with a significant risk of failure. To make it statistically feasible, I work on multiple ambitious targets. This is possible because there are no real investments associated with this research on a particular target (no expensive experimental setup): if you are stuck on one target, and another is more promising, then there is no real cost involved in switching. The targets should be unrelated so that the failure of

one does not imply failure of the other. Moreover, flexibility is key. Even if I do not hit a specific target, I often generate ideas with interesting consequences that are fully publishable, possibly at a other conferences. The feasibility depends crucially on personal intuition and experience. What are the most important challenges where I with my particular talents have a reasonable chance of making a significant difference? Most importantly for being successful in the field, am I still generating new ideas? or is it time to stop and switch to something else? Statistically the strength and feasibility is demonstrated by my CV. My three STOC/FOCS papers from 2011 span from hashing, to data structure lower bounds, to graph algorithms, demonstrating my parallel work on different directions.

The above strategy also works very well with students when I adapt the ambition level for the individual student, e.g., giving projects I am sure are feasible to beginning students. Note here that I had lots of papers even with Master's students. It has never been a problem for me recruiting students with the right kind of talent.

Conrete targets

Below I present some of the concrete targets that will be considered, starting from things I have a quite clear idea on how to approach, and moving to the more ambitious. Other problems, e.g., related to shortest paths, will also be considered. Generally I will use my broad strength to embrace the different interests of PhDs, PostDocs, and visiting collaborators.

Pseudo-random hashing This is my most practical/applied direction. Hashing is used everywhere in sequential and parallel computing, including many cases of large data sets and high volume streams where speed is essential. Conceptually a hash functions is a function mapping keys to random values, but for most applications this is impossible as it requires storing a random hash value for each possible key. Pseudo-random hashing is an area where mathematics/theory can have a major impact, proving that implementable hash functions provide important probabilistic guarantees akin to those of the impossible truly random hash functions, i.e., that they are not vulnerable to bad input distributions. A good example of the issue is linear probing which provides a very popular and efficient implementation of hash tables. Giving birth to analysis of algorithms in 1963, Knuth showed that linear probing works well with truly random hashing (in expectation and with low variance for any given input), but linear probing had a reputation of not being reliable. Indeed, if standard multiplicative hashing is used, then the input from typical DoS attacks leads to very unreliable performance [54]. In my paper [38] with Patrascu, we proved that the simplest possible tabulation hashing

provides unexpectedly strong guarantees. The scheme itself dates back more than 30 years to Carter and Wegman [55]. Keys are viewed as consisting of c characters. We initialize c tables $T_1, \ldots, T_c$ mapping characters to random hash codes. A key $x = (x_1, \ldots, x_q)$ is hashed to $T_1[x_1] \oplus \cdots \oplus T_c[x_c]$, where $\oplus$ denotes xor. Note that while we could not represent truly random tables over all keys, with 8-bit characters, we can easily store the random character tables T_i in fast cache, and that is why the scheme is very fast. However, the scheme is not even 4-independent, yet we showed that it provides many of the guarantees that are normally obtained via higher independence, e.g., Chernoff-type concentration, min-wise hashing, linear probing, and cuckoo hashing. A referee of [38] puts it nicely: *The authors of this paper are far too modest when discussing the importance of this work. At first glance tabulation hashing looks like a blunt instrument that is both inadequate (being 3- but not 4-independent) and lacking the kind of structure worthy of theoretical study. Their results contradict all these intuitions and upend a lot of conventional wisdom.* In particular, linear probing is robust with no bad input if we use simple tabulation hashing.

The project aims to provide a much better understanding of how the promise of simple randomized algorithms can be realized with simple implementable hash functions. What useful properties of truly random functions can be realized in practice? I want to emphasize that while the hash functions considered are simple, the analysis is not. Collaz' famous '$3n+1$' termination conjecture from 1937 is an extreme example of how hard it can be to analyze even the simplest algorithm. We are aiming at hashing algorithms so simple that it is like magic if they work. This follows the theory-practice tradition of Knuth, but diverges from more hard-core theory which in its focus on specific theoretical measures often sacrifices practicality. Providing a mathematical understanding of simple practical schemes, the ambitious hope is to move hashing practice to schemes not vulnerable to bad input.

A major limitation of simple tabulation hashing is that the Chernoff bounds are only for highly biased variables. This is acceptable in the above hash tables where we have $m = \Theta(n)$ bins so the probability of ending in a given bin is $1/n$, but it does not work if we use the hashing to distribute loads on a limited number of parallel machines. I am hopeful, however, that a small twist will lead to general Chernoff and Hoeffding style bounds with some of the exponential concentration that we (assuming true randomness) use everywhere in randomized algorithms and statistics. Note here that classic bounded dependence only gives polynomial concentration. Thus the hope is a simple hashing scheme providing the distributional properties that is one of the main motivations for the use of random hash functions. This will also have impact on pseudo-random number generators.

An example, not based on tabulation, is in connection with set similarity estimation via min-

hashing [10], used in data mining, clustering, machine learning, plagiarism detection, etc. There are basically two approaches to get confidence: (1) apply min-hashing k times independently, or (2) bottom-k sampling. With truly random hashing, the two options yield very similar performance. However, if we use the simplest 2-independent multiplication-shift hashing, then with (1) we may get a bias by a factor of $O(\log n)$ no matter how large k is. However, with (2), thanks to negative correlation, I am hopeful that the expected error is close to the $O(1/\sqrt{k})$ with truly random hashing. I am even hopeful that this would also work for priority sampling [16], generalizing similarity estimation to weighted sets. The difference between (1) and (2) illustrates nicely the subtlety of the area. Pseudo-random hash functions are far from random, and it is rather delicate to understand which types of randomized algorithms that can be implemented with realistic hash functions.

Deterministic dictionaries Hashing is used in all the best implementations of dictionaries, both randomized and deterministic. Chaining and linear probing allows us to support updates and queries in expected constant time [17]. Using more sophisticated methods, we know how to make the query deterministic constant time, but the updates are still randomized [22, 13]. If we want both updates and queries to be deterministic, then the best common bound known is $O(\sqrt{\log n/\log\log n})$. This uses the general dynamic predecessor search structure I devised with Andersson [3]. The bound is optimal for dynamic predecessor searching (searching nearest integer) and that is even if we allow randomization. The question is if we can do better for deterministic dictionaries. A most fundamental question is if there exist deterministic dictionaries allowing both updates and look-ups in constant time. This would be wonderful with wide consequences since a dictionary is the most basic fundamental data structure. I consider this is the most important derandomization question left in theoretical computer science.

Most likely the truth is negative in the sense that there is no perfect deterministic solution, but how do we prove it? Computer science is riddled with things we think are impossible, but where we cannot prove it, and sometimes we do find surprising solutions to seemingly impossible problems. Data structures is one of the areas where we do have matching upper and lower bounds for many problems and with Patrascu I have found some fundamental separations [39, 34]. So far, however, no one has any techniques offering such a separation between deterministic and randomized solutions. What appears the most promising strategy for a lower bound is to consider the insertion of n elements, and as in [23], divide them into epochs of exponentially increasing sizes, the smallest one being the most recent. We would hope to prove that a look-up algorithm has to make something

in the style of a binary search to find out which epoch a key was inserted in, if any. This kind of argument would lead to an $\Omega(\log\log n)$ lower bound, and is consistent with known upper-bound techniques. On the upper-bound side there are several related questions to consider, e.g., improving the update time for deterministic dynamic predecessor search.

Graph algorithms Recall that I deliberately work in parallel on different directions, always looking for techniques to make a significant difference on important algorithmic problems. Within graph algorithms, I will study the coloring of 3-colorable graphs which is my all-time favorite approximation problem. Contrasting the easy 2-coloring, it is often the first NP-hard problem students are taught. The question is how well we can color 3-colorable graphs in polynomial time, and it has engaged many of the most famous theoretical computer scientists [5, 8, 9, 12, 14, 28, 56]. I will involve Ken-Ichi Kawarabayashi (Tokyo, Japan) whose strength is on the graph theory side of graph algorithms, complementing my own more algorithmic background. We will also involve Carsten Thomassen (DTU, Denmark) who is an expert on the pure graph theory side of coloring. Nobody has been able to do anything combinatorially about the problem since Blum at FOCS'90 [8]. Semi-definite programming (SDP) has been applied and exhausted, and the current best results are obtained balancing the combinatorial and the SDP approaches. Our target is to improve things on the combinatorial side, developing a stronger understanding of the nature of 3-colorable graphs. Given how many famous people had worked on the problem, I thought this would be the most unrealistic target, but Ken-Ichi and I may already have our first progress: a way to witness certain monochromatic sets in 3-colorable graphs based on bipartite expanders. This should lead to the first combinatorial improvement in 22 years, and also improve the overall bounds when combined with the latest SDP. The big dream, however, is to get a subpolynomial approximation factor.

We expect to work on several other approximation problems in graph algorithms, e.g., the long-standing $2-o(1)$ factor for k-way cut (the problems we proved fixed-parameter tractable in [29]). It happens that I do have the best approximation factor known for the similar-looking k-terminal cut problem [27], but that was using linear programming. Here the general idea is to bring in more combinatorial understanding, complementing all the recent work based on linear and semi-definite programming.

Milestones The concept of milestones is rather peculiar when you are looking for surprising ideas. However, as intermediate goals in efficient algorithms, you just have to find a new way, allowing you

to do better than anyone else. The more significant and interesting the improvement, the stronger a venue you publish in. A good example is in connection with 3-coloring where Ken and I already think we see a path to the first combinatorial improvement in 22 years. Even if this would not close the problem, I do expect that such a result would steer interest at STOC/FOCS. We can view as milestones all targets above for which I stated that I am already hopeful about a way. Another view is that many interesting discoveries are expected throughout the project. Except for first year PhD students, I expect each member to be part of 1–3 STOC/FOCS/SODA level discoveries a year.

The Danish perspective

Denmark has already committed strongly to algorithms and the theory of computing, particularly at Århus University with no less than three impressive centers of excellence from the Danish National Research Foundation (BRICS,MADALGO,and CTIC). We have Lars Arge from Århus with the Danish Elite Research award from 2010 and Rasmus Pagh from ITU with a Danish Sapere Aude Starting Grant from 2011. Denmark has a very strong standing in algorithms in Europe, only dominated by Israel. My return to Denmark would nevertheless bring in a new level of international strength, visibility, and recognition in fundamental algorithms and data structures.

As mentioned in my CV, I am the only one in the world who is on the editorial board of all the top three journals for algorithmic research. In particular I am area editor of "algorithms and data structures" for ACM's flagship scientific journal *J. ACM*. No one else in Denmark is on any of them, but Arge is on the editorial board of the fourth choice *Algorithmica*. Likewise, when we look at the top theory conferences STOC/FOCS, I have been on 9 PCs. This is almost twice that of the rest of Denmark combined. The main reason for my strong presence on these editorial boards and PCs is my research. I have 26 STOC/FOCS papers which is nearly thrice that of anyone else in Denmark. Peter Bro Miltersen from Århus is second with 9 STOC/FOCS papers. At the top algorithms conference SODA I have 23 papers while Gerth Brodal from Århus is second with 10 SODA papers. These are the strongest venues from general algorithmic perspective (see, e.g., `www.en.wikipedia.org/wiki/List_of_computer_science_conferences`). My h-factor [Google Scholar] is 44. Within algorithms in Denmark, Lars Arge is second at 34. I am Fellow of the ACM in algorithms and data structures. In Denmark the only other ones with this highest ACM rank in computer science are Christian Søndergaard Jensen from Århus in databases, Neil Jones retired from DIKU in programming languages, and Dines Bjørner retired from DTU in formal methods.

The focuses of the centers in Århus are different. MADALGO is focused on geometric algorithms and external memory hierarchies (this shows, e.g., in Arge's strong record in computational geometry with 10 SoCG papers) whereas I work on more general algorithmic issues. The connection between MADALGO and Christian S. Jensen in spatial data bases is going to be very interesting. CTIC is further away with its focus on complexity, cryptography, and games (other subareas of theoretical computer science covered by STOC/FOCS). Nevertheless there are many interesting connections, and I expect many fruitful collaborations with Århus, e.g., we need a subquadratic algorithm to even consider data so large that the external memory is needed. With my interest in hashing and hash tables I overlap more in interests with Rasmus Pagh and together we would form the strongest group for this in the world. At some stage it would be obvious for us join forces and create a bigger center together.

Mathematically my main connection and inspiration in Denmark is Carsten Thomassen from DTU. When I was an undergraduate at DTU, he advised me on how to approach famous problems that have defeated many researchers; namely to work flexibly on several such high gain targets in parallel, switching between them and following your inspiration so that a problem that is too hard does not stop you. This organic approach to targets beyond planning has lead both of us to many breakthroughs. Combining his back ground in graph theory with my more algorithmic back ground will yield a strong force in algorithmic graph theory where we would also bring in Ken-Ichi Kawarabayashi from Japan.

I hope to return as a uniting figure for algorithms in Denmark. Despite being mostly out of the country, I have collaborated with many Danish algorithms researchers, e.g., Miltersen (Århus) in [2], Pagh (ITU) in [32], Alstrup (ITU), Rauhe (ITU), and Gørtz (DTU) in [1], Bille (DTU) in [7], and I have a paper in the workings with Arge (Århus). My international collaborators will give seminars and broadly boost algorithmic activity in the region, but this requires a solid visitor budget to invite them. Otherwise I will be the visitor touring the world with expenses paid by my collaborators, and with much less benefit to Denmark.

I want to emphasize that compared with the centers in Århus, what I apply for now is small. Think of it as the difference between a big party where the host runs around making sure everyone is happy, and a smaller event with a few friends where you really get to talk. Århus has done an amazing job with many PostDocs and great events. My talent is for research and to collaborate, inspire, and guide on a more personal level, using my broad strength and experience within algorithms and data structures. This is why I only want a (much cheaper) selective mini-center where I can be more

directly involved in the research activities (less manager, more leader).

Project members

I would devote 50% of my total work time exclusively to this project at the University of Copenhagen. Furthermore I would spend 25% of my time on regular teaching, adding up to at least 75% of my total time in Copenhagen with the vision of re-establishing myself permanently. The project includes a PhD student at all times: one in years 1–3, and one in years 4–6 (last year paid by the department). Moreover the project includes a Post Doc in years 2–3. Except for first year PhD students, I expect each member to be part of 1–3 STOC/FOCS/SODA level discoveries a year. Finally, to establish a new focal point in the area, it is desirable to have a changing prominent visitor for collaboration and inspiration. The exact cost will depend on the visitor. Sometimes we will have long term visitors that need some salary, e.g., a US professor on sabbatical with 70% pay from home institution. At other times, we will have short term visitors needing no salary, but more travel expenses. Expected visitors are current collaborators like Kawarabayachi, Patrascu, Zwick, but generally I will aim to invite those behind the most exiting new developments related to our research.

Related to the project, I am heading the hiring a new tenure-track assistant professor in algorithms at the University of Copenhagen. I will also have a PhD student from the department. Finally, when we have had some years to get established, I hope to get alternative funding for a Post Doc for years 4–5.

References

[1] S. Alstrup, T. Rauhe, I.L. Gørtz, M. Thorup, and U. Zwick. Union-find with constant time deletions. *Proc. 32th ICALP*, pages 78–89, 2005.

[2] A. Andersson, P. B. Miltersen, S. Riis, and M. Thorup. Static dictionaries on AC^0 RAMs: Query time $\Theta(\sqrt{\log n/\log\log n})$ is necessary and sufficient. In *Proc. 37th FOCS*, pages 441–450, 1996.

[3] A. Andersson and M. Thorup. Dynamic ordered sets with exponential search trees. *J. ACM*, 54(3):Article 13, 2007.

[4] R. Agarwala, V. Bafna, M. Farach, B. Narayanan, M. Paterson, and M. Thorup. On the approximability of numerical taxonomy (fitting distances by tree metrics). *SIAM J. Comput.*, 28(3):1073 – 1085, 1999.

[5] S. Arora, E. Chlamtac, and M. Charikar. New approximation guarantee for chromatic number. *Proc. 38th STOC*, 215–224, 2006.

[6] S. Arora and R. Ge, New Tools for Graph Coloring. *Proc. 14th APPROX-RANDOM*, pages 1–12, 2011.

[7] P. Bille and M. Thorup. Regular expression matching with multi-strings and intervals. *Proc. 21st SODA*, pages 1279–1308, 2010.

[8] A. Blum. New Approximation Algorithms for Graph Coloring. *J. ACM*, 41(3): 470–516, 1994.

[9] A. Blum and D.R. Karger. An $\tilde{O}(n^{3/14})$-Coloring Algorithm for 3-Colorable Graphs. *Inf. Process. Lett.* 61(1): 49–53, 1997.

[10] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *J. Comput. System Sci.*, 60(3):630–659, 2000.

[11] A. L. Buchsbaum, H. J. Karloff, C. Kenyon, N. Reingold, and M. Thorup. OPT versus LOAD in dynamic storage allocation. *SIAM J. Comput.*, 33(3):632–646, 2004.

[12] E. Chlamtac. Approximation Algorithms Using Hierarchies of Semidefinite Programming Relaxations. *Proc. 48th FOCS*, 687–696, 2007.

[13] M. Dietzfelbinger, A.R. Karlin, K. Mehlhorn, F.M. Heide, H. Rohnert, and R.E. Tarjan. Dynamic Perfect Hashing: Upper and Lower Bounds. *SIAM J. Comput.*, 23(4):738–761, 1994.

[14] I. Dinur, E. Mossel, and O. Regev. Conditional Hardness for Approximate Coloring. *SIAM J. Computing*, 39(3):843–873, 2009.

[15] N. Duffield, C. Lund, and M. Thorup. Estimating flow distributions from sampled flow statistics. *ACM/IEEE Trans. Networking*, 13(5):933–946, 2005. Invited from SIGCOMM'03.

[16] N. Duffield, C. Lund, and M. Thorup. Priority sampling for estimation of arbitrary subset sums. *J. ACM*, 54(6):Article 32, 2007.

[17] A. I. Dumey. Indexing for rapid random access memory systems. *Computers and Automation*, 5(12):6–9, 1956.

[18] R.W. Floyd. Algorithm 97: Shortest Path. *Comm. ACM*, 5 (6): 345, 1962.

[19] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, 40(10):118–124, 2002.

[20] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proc. 19th INFOCOM*, pages 519–528, 2000. Partly covered by [21].

[21] B. Fortz and M. Thorup. Increasing Internet capacity using local search. *Computational Optimization and Applications*, 29(1):13–48, 2004.

[22] M.L. Fredman, J. Komlós and E. Szemerédi. Storing a Sparse Table with $0(1)$ Worst Case Access Time. *J. ACM*, 31(3):538–544, 1984.

[23] M.L. Fredman and M.E. Saks. The cell probe complexity of dynamic data structures. *Proc. 21st STOC*, pages 345–354, 1989.

[24] T. Hagerup, P.B. Miltersen, R. Pagh Deterministic Dictionaries. *J. Algorithms* 41(1):69-85 2001.

[25] Y. Han and M. Thorup. Integer sorting in $O(n\sqrt{\log\log n})$ expected time and linear space. In *Proc. 43nd FOCS*, pages 135–144, 2002.

[26] J. Holm, K. Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge and biconnectivity. *J. ACM*, 48(4):723–760, 2001.

[27] D. Karger, P. Klein, C. Stein, , M. Thorup, and N. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. *Math. Oper. Res.*, 29(3):436–461, 2004.

[28] D.R. Karger, R. Motwani, M. Sudan. Approximate Graph Coloring by Semidefinite Programming. *J. ACM*, 45(2):246–265 (1998)

[29] K. Kawarabayashi and M. Thorup. Minimum *k*-way cut of bounded size is fixed-parameter tractable, *Proc. 52nd FOCS*, pages 160–169, 2011.

[30] R. Pagh. A Trade-Off for Worst-Case Efficient Dictionaries. *Nord.J. Comput.* 7(3): 151-163 (2000)

[31] A. Pagh and R. Pagh. Uniform Hashing in Constant Time and Optimal Space. *SIAM J. Comput.* 38(1): 85-96 (2008)

[32] A. Pagh, R. Pagh, and M. Thorup. On adaptive integer sorting. *Proc. 12th ESA*, pages 556–579, 2004.

[33] M. Paterson, Y. Peres, M. Thorup, P. Winkler, and U. Zwick. Maximum overhang. *The American Mathematical Monthly*, 116(9):763–787, 2009. Featured in *Science*, page 323: 875, Feb. 13, 2009. Co-winner of MAA Robbins Award 2011.

[34] M. Pătraşcu and M. Thorup. Time-space trade-offs for predecessor search. In *Proc. 38th STOC*, pages 232–240, 2006.

[35] M. Pătraşcu and M. Thorup. Randomization does not help searching predecessors. In *Proc. 18th SODA*, pages 555–564, 2007.

[36] M. Pătraşcu and M. Thorup. Planning for fast connectivity updates. In *Proc. 48th FOCS*, pages 646–654, 2007.

[37] M. Patrascu and M. Thorup. On the independence required by linear probing and minwise independence. In *Proc. 36th ICALP*, pages 715–726, 2010.

[38] M. Pătraşcu and M. Thorup. The power of simple tabulation hashing. *Proc. 43rd STOC*, pages 1–10, 2011. Full version `http://arxiv.org/abs/1011.5200` will appear in *J. ACM*.

[39] M. Pătraşcu and M. Thorup. Don't rush into a union: take time to find your roots. *Proc. 43rd STOC*, pages 559–568, 2011.

[40] A. Siegel. On universal classes of extremely random constant-time hash functions. *SIAM J. Computing*, 33(3):505–543, 2004. Announced at FOCS'89.

[41] J.P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics*, 8(2):223–250, 1995. See also SODA'93.

[42] C. Thomassen. Every planar graph is 5-choosable. *J. Comb. Theory, Ser. B*, 62(1):180–181, 1994.

[43] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004.

[44] M. Thorup. Fully-dynamic min-cut. *Combinatorica*, 27(1):91–127, 2007.

[45] M. Thorup. Minimum k-way cuts via deterministic greedy tree packing. *Proc. 40th STOC*, pages 159–166, 2008.

[46] M. Thorup. Combinatorial power in multimedia processors. *ACM SIGARCH Computer Architecture News*, 31(5):5–11, 2003.

[47] M. Thorup and U. Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005.

[48] M. Thorup. Equivalence between priority queues and sorting. *J. ACM*, 54(6):Article 28, 2007.

[49] M. Thorup. Map graphs in polynomial time. In *Proc. 39th FOCS*, pages 396–405, 1998.

[50] M. Thorup. Undirected single source shortest paths with positive integer weights in linear time. *J. ACM*, 46(3):362–394, 1999.

[51] M. Thorup. Integer priority queues with decrease key in constant time and the single source shortest paths problem. *J. Comput. Syst. Sc.*, 69(3):330–353, 2004.

[52] M. Thorup. String hashing for linear probing. In *Proc. 20th SODA*, pages 655–664, 2009.

[53] M. Thorup. Timeouts with Time-Reversed Linear Probing. *Proc. 30th INFOCOM*, pages 166–170, 2011.

[54] M. Thorup and Y. Zhang. Tabulation based 5-universal hashing and linear probing. *Proc. 12th ALENEX*, pages 62–76, 2010.

[55] M.N. Wegman and L. Carter: New Hash Functions and Their Use in Authentication and Set Equality. *J. Comput. Syst. Sci.* 22(3): 265-279 (1981). Announced at FOCS'79.

[56] A. Wigderson. Improving the Performance Guarantee for Approximate Graph Coloring *J. ACM*, 30(4):729-735, 1983.