Juraj Hromkovič
Richard Královič
Marc Nunkesser
Peter Widmayer (Eds.)

LNCS 4665

# Stochastic Algorithms: Foundations and Applications

4th International Symposium, SAGA 2007
Zurich, Switzerland, September 2007
Proceedings



Springer

Juraj Hromkovič Richard Královič
Marc Nunkesser Peter Widmayer (Eds.)

# Stochastic Algorithms: Foundations and Applications

4th International Symposium, SAGA 2007
Zurich, Switzerland, September 13-14, 2007
Proceedings

Springer

Volume Editors

Juraj Hromkovič
Richard Královič
Marc Nunkesser
Peter Widmayer
Swiss Federal Institute of Technology
Department of Computer Science ETH Zentrum
8092 Zürich, Switzerland
E-mail: {juraj.hromkovic,richard.kralovic,mnunkess,widmayer}@inf.ethz.ch

Library of Congress Control Number: 2007934297

CR Subject Classification (1998): F.2, F.1.2, G.1.2, G.1.6, G.2, G.3

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

| | |
|---|---|
| ISSN | 0302-9743 |
| ISBN-10 | 3-540-74870-9 Springer Berlin Heidelberg New York |
| ISBN-13 | 978-3-540-74870-0 Springer Berlin Heidelberg New York |



Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12121038 06/3180 5 4 3 2 1 0

Lecture Notes in Computer Science 4665

*Commenced Publication in 1973*
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

# Lecture Notes in Computer Science

Sublibrary 1: Theoretical Computer Science and General Issues

For information about Vols. 1– 4422
please contact your bookseller or Springer

Vol. 4743: P. Thulasiraman, X. He, T.L. Xu, M.K. Denko, R.K. Thulasiram, L.T. Yang (Eds.), Frontiers of High Performance Computing and Networking ISPA 2007 Workshops. XXIX, 536 pages. 2007.

Vol. 4742: I. Stojmenovic, R.K. Thulasiram, L.T. Yang, W. Jia, M. Guo, R.F. de Mello (Eds.), Parallel and Distributed Processing and Applications. XX, 995 pages. 2007.

Vol. 4736: S. Winter, M. Duckham, L. Kulik, B. Kuipers (Eds.), Spatial Information Theory. XV, 455 pages. 2007.

Vol. 4732: K. Schneider, J. Brandt (Eds.), Theorem Proving in Higher Order Logics. IX, 401 pages. 2007.

Vol. 4708: L. Kučera, A. Kučera (Eds.), Mathematical Foundations of Computer Science 2007. XVIII, 764 pages. 2007.

Vol. 4707: O. Gervasi, M.L. Gavrilova (Eds.), Computational Science and Its Applications – ICCSA 2007, Part III. XXIV, 1205 pages. 2007.

Vol. 4706: O. Gervasi, M.L. Gavrilova (Eds.), Computational Science and Its Applications – ICCSA 2007, Part II. XXIII, 1129 pages. 2007.

Vol. 4705: O. Gervasi, M.L. Gavrilova (Eds.), Computational Science and Its Applications – ICCSA 2007, Part I. XLIV, 1169 pages. 2007.

Vol. 4703: L. Caires, V.T. Vasconcelos (Eds.), CONCUR 2007 – Concurrency Theory. XIII, 507 pages. 2007.

Vol. 4697: L. Choi, Y. Paek, S. Cho (Eds.), Advances in Computer Systems Architecture. XIII, 400 pages. 2007.

Vol. 4688: K. Li, M. Fei, G.W. Irwin, S. Ma (Eds.), Bio-Inspired Computational Intelligence and Applications. XIX, 805 pages. 2007.

Vol. 4684: L. Kang, Y. Liu, S. Zeng (Eds.), Evolvable Systems: From Biology to Hardware. XIV, 446 pages. 2007.

Vol. 4683: L. Kang, Y. Liu, S. Zeng (Eds.), Intelligence Computation and Applications. XVII, 663 pages. 2007.

Vol. 4681: D.-S. Huang, L. Heutte, M. Loog (Eds.), Advanced Intelligent Computing Theories and Applications. XXVI, 1379 pages. 2007.

Vol. 4671: V. Malyshkin (Ed.), Parallel Computing Technologies. XIV, 635 pages. 2007.

Vol. 4669: J.M. de Sá, L.A. Alexandre, W. Duch, D. Mandic (Eds.), Artificial Neural Networks – ICANN 2007, Part II. XXXI, 990 pages. 2007.

Vol. 4668: J.M. de Sá, L.A. Alexandre, W. Duch, D. Mandic (Eds.), Artificial Neural Networks – ICANN 2007, Part I. XXXI, 978 pages. 2007.

Vol. 4666: M.E. Davies, C.J. James, S.A. Abdallah, M.D Plumbley (Eds.), Independent Component Analysis and Blind Signal Separation. XIX, 847 pages. 2007.

Vol. 4665: J. Hromkovič, R. Královič, M. Nunkesser, P. Widmayer (Eds.), Stochastic Algorithms: Foundations and Applications. X, 167 pages. 2007.

Vol. 4664: J. Durand-Lose, M. Margenstern (Eds.), Machines, Computations, and Universality. X, 325 pages. 2007.

Vol. 4649: V. Diekert, M.V. Volkov, A. Voronkov (Eds.), Computer Science – Theory and Applications. XIII, 420 pages. 2007.

Vol. 4647: R. Martin, M. Sabin, J. Winkler (Eds.), Mathematics of Surfaces XII. IX, 509 pages. 2007.

Vol. 4644: N. Azemard, L. Svensson (Eds.), Integrated Circuit and System Design. XIV, 583 pages. 2007.

Vol. 4641: A.-M. Kermarrec, L. Bougé, T. Priol (Eds.), Euro-Par 2007 Parallel Processing. XXVII, 974 pages. 2007.

Vol. 4639: E. Csuhaj-Varjú, Z. Ésik (Eds.), Fundamentals of Computation Theory. XIV, 508 pages. 2007.

Vol. 4638: T. Stützle, M. Birattari, H.H. Hoos (Eds.), Engineering Stochastic Local Search Algorithms. X, 223 pages. 2007.

Vol. 4628: L.N. de Castro, F.J. Von Zuben, H. Knidel (Eds.), Artificial Immune Systems. XII, 438 pages. 2007.

Vol. 4627: M. Charikar, K. Jansen, O. Reingold, J.D.P. Rolim (Eds.), Approximation, Randomization, and Combinatorial Optimization. XII, 626 pages. 2007.

Vol. 4624: T. Mossakowski, U. Montanari, M. Haveraaen (Eds.), Algebra and Coalgebra in Computer Science. XI, 463 pages. 2007.

Vol. 4619: F. Dehne, J.-R. Sack, N. Zeh (Eds.), Algorithms and Data Structures. XVI, 662 pages. 2007.

Vol. 4618: S.G. Akl, C.S. Calude, M.J. Dinneen, G. Rozenberg, H.T. Wareham (Eds.), Unconventional Computation. X, 243 pages. 2007.

Vol. 4616: A. Dress, Y. Xu, B. Zhu (Eds.), Combinatorial Optimization and Applications. XI, 390 pages. 2007.

Vol. 4613: F.P. Preparata, Q. Fang (Eds.), Frontiers in Algorithmics. XI, 348 pages. 2007.

Vol. 4600: H. Comon-Lundh, C. Kirchner, H. Kirchner (Eds.), Rewriting, Computation and Proof. XVI, 273 pages. 2007.

Vol. 4599: S. Vassiliadis, M. Berekovic, T.D. Hämäläinen (Eds.), Embedded Computer Systems: Architectures, Modeling, and Simulation. XVIII, 466 pages. 2007.

Vol. 4598: G. Lin (Ed.), Computing and Combinatorics. XII, 570 pages. 2007.

Vol. 4596: L. Arge, C. Cachin, T. Jurdziński, A. Tarlecki (Eds.), Automata, Languages and Programming. XVII, 953 pages. 2007.

Vol. 4595: D. Bošnački, S. Edelkamp (Eds.), Model Checking Software. X, 285 pages. 2007.

Vol. 4590: W. Damm, H. Hermanns (Eds.), Computer Aided Verification. XV, 562 pages. 2007.

Vol. 4588: T. Harju, J. Karhumäki, A. Lepistö (Eds.), Developments in Language Theory. XI, 423 pages. 2007.

Vol. 4583: S.R. Della Rocca (Ed.), Typed Lambda Calculi and Applications. X, 397 pages. 2007.

Vol. 4580: B. Ma, K. Zhang (Eds.), Combinatorial Pattern Matching. XII, 366 pages. 2007.

Vol. 4576: D. Leivant, R. de Queiroz (Eds.), Logic, Language, Information and Computation. X, 363 pages. 2007.

Vol. 4547: C. Carlet, B. Sunar (Eds.), Arithmetic of Finite Fields. XI, 355 pages. 2007.

Vol. 4546: J. Kleijn, A. Yakovlev (Eds.), Petri Nets and Other Models of Concurrency – ICATPN 2007. XI, 515 pages. 2007.

Vol. 4545: H. Anai, K. Horimoto, T. Kutsia (Eds.), Algebraic Biology. XIII, 379 pages. 2007.

Vol. 4533: F. Baader (Ed.), Term Rewriting and Applications. XII, 419 pages. 2007.

Vol. 4528: J. Mira, J.R. Álvarez (Eds.), Nature Inspired Problem-Solving Methods in Knowledge Engineering, Part II. XXII, 650 pages. 2007.

Vol. 4527: J. Mira, J.R. Álvarez (Eds.), Bio-inspired Modeling of Cognitive Tasks, Part I. XXII, 630 pages. 2007.

Vol. 4525: C. Demetrescu (Ed.), Experimental Algorithms. XIII, 448 pages. 2007.

Vol. 4514: S.N. Artemov, A. Nerode (Eds.), Logical Foundations of Computer Science. XI, 513 pages. 2007.

Vol. 4513: M. Fischetti, D.P. Williamson (Eds.), Integer Programming and Combinatorial Optimization. IX, 500 pages. 2007.

Vol. 4510: P. Van Hentenryck, L.A. Wolsey (Eds.), Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. X, 391 pages. 2007.

Vol. 4507: F. Sandoval, A. Prieto, J. Cabestany, M. Graña (Eds.), Computational and Ambient Intelligence. XXVI, 1167 pages. 2007.

Vol. 4501: J. Marques-Silva, K.A. Sakallah (Eds.), Theory and Applications of Satisfiability Testing – SAT 2007. XI, 384 pages. 2007.

Vol. 4497: S.B. Cooper, B. Löwe, A. Sorbi (Eds.), Computation and Logic in the Real World. XVIII, 826 pages. 2007.

Vol. 4494: H. Jin, O.F. Rana, Y. Pan, V.K. Prasanna (Eds.), Algorithms and Architectures for Parallel Processing. XIV, 508 pages. 2007.

Vol. 4493: D. Liu, S. Fei, Z. Hou, H. Zhang, C. Sun (Eds.), Advances in Neural Networks – ISNN 2007, Part III. XXVI, 1215 pages. 2007.

Vol. 4492: D. Liu, S. Fei, Z. Hou, H. Zhang, C. Sun (Eds.), Advances in Neural Networks – ISNN 2007, Part II. XXVII, 1321 pages. 2007.

Vol. 4491: D. Liu, S. Fei, Z.-G. Hou, H. Zhang, C. Sun (Eds.), Advances in Neural Networks – ISNN 2007, Part I. LIV, 1365 pages. 2007.

Vol. 4490: Y. Shi, G.D. van Albada, J. Dongarra, P.M.A. Sloot (Eds.), Computational Science – ICCS 2007, Part IV. XXXVII, 1211 pages. 2007.

Vol. 4489: Y. Shi, G.D. van Albada, J. Dongarra, P.M.A. Sloot (Eds.), Computational Science – ICCS 2007, Part III. XXXVII, 1257 pages. 2007.

Vol. 4488: Y. Shi, G.D. van Albada, J. Dongarra, P.M.A. Sloot (Eds.), Computational Science – ICCS 2007, Part II. XXXV, 1251 pages. 2007.

Vol. 4487: Y. Shi, G.D. van Albada, J. Dongarra, P.M.A. Sloot (Eds.), Computational Science – ICCS 2007, Part I. LXXXI, 1275 pages. 2007.

Vol. 4484: J.-Y. Cai, S.B. Cooper, H. Zhu (Eds.), Theory and Applications of Models of Computation. XIII, 772 pages. 2007.

Vol. 4475: P. Crescenzi, G. Prencipe, G. Pucci (Eds.), Fun with Algorithms. X, 273 pages. 2007.

Vol. 4474: G. Prencipe, S. Zaks (Eds.), Structural Information and Communication Complexity. XI, 342 pages. 2007.

Vol. 4459: C. Cérin, K.-C. Li (Eds.), Advances in Grid and Pervasive Computing. XVI, 759 pages. 2007.

Vol. 4449: Z. Horváth, V. Zsók, A. Butterfield (Eds.), Implementation and Application of Functional Languages. X, 271 pages. 2007.

Vol. 4448: M. Giacobini (Ed.), Applications of Evolutionary Computing. XXIII, 755 pages. 2007.

Vol. 4447: E. Marchiori, J.H. Moore, J.C. Rajapakse (Eds.), Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. XI, 302 pages. 2007.

Vol. 4446: C. Cotta, J. van Hemert (Eds.), Evolutionary Computation in Combinatorial Optimization. XII, 241 pages. 2007.

Vol. 4445: M. Ebner, M. O'Neill, A. Ekárt, L. Vanneschi, A.I. Esparcia-Alcázar (Eds.), Genetic Programming. XI, 382 pages. 2007.

Vol. 4436: C.R. Stephens, M. Toussaint, D. Whitley, P.F. Stadler (Eds.), Foundations of Genetic Algorithms. IX, 213 pages. 2007.

Vol. 4433: E. Şahin, W.M. Spears, A.F.T. Winfield (Eds.), Swarm Robotics. XII, 221 pages. 2007.

Vol. 4432: B. Beliczynski, A. Dzielinski, M. Iwanowski, B. Ribeiro (Eds.), Adaptive and Natural Computing Algorithms, Part II. XXVI, 761 pages. 2007.

Vol. 4431: B. Beliczynski, A. Dzielinski, M. Iwanowski, B. Ribeiro (Eds.), Adaptive and Natural Computing Algorithms, Part I. XXV, 851 pages. 2007.

Vol. 4424: O. Grumberg, M. Huth (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. XX, 738 pages. 2007.

Vol. 4423: H. Seidl (Ed.), Foundations of Software Science and Computational Structures. XVI, 379 pages. 2007.

# Preface

The 4th Symposium on Stochastic Algorithms, Foundations and Applications (SAGA 2007) took place on September 13–14, 2007, in Zürich (Switzerland). It offered the opportunity to present original research on the design and analysis of randomized algorithms, complexity theory of randomized computations, random combinatorial structures, implementation, experimental evaluation and real-world application of stochastic algorithms. In particular, the focus of the SAGA symposia series is on investigating the power of randomization in algorithmics, and on the theory of stochastic processes especially within realistic scenarios and applications. Thus, the scope of the symposium ranges from the study of theoretical fundamentals of randomized computation to experimental algorithmics related to stochastic approaches.

Previous SAGA symposia took place in Berlin (2001), Hatfield (2003), and Moscow (2005). This year 31 submissions were received, and the program committee selected 9 submissions for presentation. All papers were evaluated by at least four members of the program committee, partly with the assistance of subreferees. We thank the members of the program committee as well as all subreferees for their thorough and careful work. A special thanks goes to Harry Buhrman, Martin Dietzfelbinger, Rūsiņš Freivalds, Paul G. Spirakis, and Aravind Srinivasan, who accepted our invitation to give invited talks at SAGA 2007 and so to share their insights on new developments in research areas of key interest.

September 2007

Juraj Hromkovič
Richard Královič
Marc Nunkesser
Peter Widmayer

# Organization

## Program Chairs

Juraj Hromkovič
Peter Widmayer

## Program Committee

Farid Ablayev
Dimitris Achlioptas
Andreas A. Albrecht
Andris Ambainis
Eli Ben-Sasson
Markus Bläser
Harry Buhrman
Colin Cooper
Josep Diaz
Martin Dietzfelbinger
Oktay M. Kasim-Zade
C. Pandu Rangan
Vijaya Ramachandran
Jose Rolim
Vishal Sanwalani
Martin Sauerhoff
Christian Scheideler
Georg Schnitger
Jiri Sgall
Angelika Steger
Kathleen Steinhöfel
Berthold Vöcking
Osamu Watanabe

## Local Organization

Richard Královič
Tobias Mömke
Marc Nunkesser

## External Reviewers

Peter Bosman
Alex Fukunaga
Aida Gainutdinova
Peter vd Gulik
Florian Jug
Fabian Kuhn
Julian Lorenz
Conrado Martinez
Dieter Mitsche
Jan Remy
Robert Spalek
Dirk Sudholt
Falk Unger
Nikolai Vereshchagin
Andrey Voronenko

# Table of Contents

## Invited Papers

On Computation and Communication with Small Bias ..... 1
*Harry Buhrman*

Design Strategies for Minimal Perfect Hash Functions ..... 2
*Martin Dietzfelbinger*

Hamming, Permutations and Automata ..... 18
*Rūsiņš Freivalds*

Probabilistic Techniques in Algorithmic Game Theory ..... 30
*Spyros C. Kontogiannis and Paul G. Spirakis*

Randomized Algorithms and Probabilistic Analysis in Wireless Networking ..... 54
*Aravind Srinivasan*

## Contributed Papers

A First Step Towards Analyzing the Convergence Time in Player-Specific Singleton Congestion Games ..... 58
*Heiner Ackermann*

Communication Problems in Random Line-of-Sight Ad-Hoc Radio Networks ..... 70
*Artur Czumaj and Xin Wang*

Approximate Discovery of Random Graphs ..... 82
*Thomas Erlebach, Alexander Hall, and Matúš Mihal'ák*

A VNS Algorithm for Noisy Problems and Its Application to Project Portfolio Analysis ..... 93
*Walter J. Gutjahr, Stefan Katzensteiner, and Peter Reiter*

Digit Set Randomization in Elliptic Curve Cryptography ..... 105
*David Jao, S. Ramesh Raju, and Ramarathnam Venkatesan*

Lower Bounds for Hit-and-Run Direct Search ..... 118
*Jens Jägersküpper*

An Exponential Gap Between LasVegas and Deterministic Sweeping Finite Automata ..... 130
*Christos Kapoutsis, Richard Královič, and Tobias Mömke*

Stochastic Methods for Dynamic OVSF Code Assignment in 3G Networks ..... 142
*Mustafa Karakoc and Adnan Kavak*

On the Support Size of Stable Strategies in Random Games ..... 154
*Spyros C. Kontogiannis and Paul G. Spirakis*

**Author Index** ..... 167

# On Computation and Communication with Small Bias

Harry Buhrman

Centrum voor Wiskunde en Informatica (CWI) & University of Amsterdam
The Netherlands

**Abstract.** Many models in theoretical computer science allow for computations or representations where the answer is only slightly biased in the right direction. The best-known of these is the complexity class PP, for "probabilistic polynomial time". A language is in PP if there is a randomized polynomial-time Turing machine whose acceptance probability is greater than 1/2 if, and only if, its input is in the language.

Most computational complexity classes have an analogous class in communication complexity. The class PP in fact has two, a version with weakly restricted bias called PPcc, and a version with unrestricted bias called UPPcc. Ever since their introduction by Babai, Frankl, and Simon in 1986, it has been open whether these classes are the same. We show that PPcc is strictly included in UPPcc. Our proof combines a query complexity separation due to Beigel with a technique of Razborov that translates the acceptance probability of quantum protocols to polynomials. We will discuss some complexity theoretical consequences of this separation. This presentation is bases on joined work with Nikolay Vereshchagin and Ronald de Wolf.

J. Hromkovič et al. (Eds.): SAGA 2007, LNCS 4665, p. 1, 2007.

# Design Strategies for Minimal Perfect Hash Functions

Martin Dietzfelbinger

Technische Universität Ilmenau, 98684 Ilmenau, Germany
martin.dietzfelbinger@tu-ilmenau.de

**Abstract.** A minimal perfect hash function $h$ for a set $S \subseteq U$ of size $n$ is a function $h: U \to \{0, \ldots, n-1\}$ that is one-to-one on $S$. The complexity measures of interest are storage space for $h$, evaluation time (which should be constant), and construction time. The talk gives an overview of several recent randomized constructions of minimal perfect hash functions, leading to space-efficient solutions that are fast in practice. A central issue is a method ("split-and-share") that makes it possible to assume that fully random (hash) functions are available.

## 1 Introduction

In this survey paper we discuss algorithmic techniques that are useful for the construction of minimal perfect hash functions. We focus on techniques for managing randomness.

We assume a set $U = \{0,1\}^w$ (the "universe") of "keys" $x$ is given. Assume that $S \subseteq U$ is a (given) set with cardinality $n = |S|$, and that $m \geq n$. A function $h: U \to [m]$ that is one-to-one on $S$ is called a *perfect hash function* (for $S$). If in addition $n = m$ (the smallest possible value), $h$ is called a *minimal perfect hash function* (MPHF).[1]

The MPHF problem for a given $S \subseteq U$ is to construct a data structure $D_h$ that allows us to evaluate $h(x)$ for given $x \in U$, where $h$ is a MPHF for $S$. The parameters of interest are the storage space for $D_h$ and the evaluation time of $h$, which should be constant. Clearly, such a data structure $D_h$ can be used to devise a (static) dictionary that for each key $x \in S$ stores $x$ and some data item $d_x$ in an array of size $n$, with constant retrieval time.

In the past decades, the MPHF problem has been studied thoroughly. For a detailed survey of the developments up to 1997 see the comprehensive study [9]. To put the results into perspective, one should notice the fundamental space lower bound of $n \log e + \log w - O(\log n)$ bits[2], valid as soon as $w \geq (2 + \varepsilon) \log n$, proved by Fredman and Komlós [18]. This bound is essentially tight: Mehlhorn [23, Sect. III.2.3, Thm. 8] gave a construction of a MPHF that takes $n \log e + \log w + O(\log n)$ bits of space (but has a vast evaluation time). In order not to have to worry about the influence of the size $2^w$ of $U$ too much, unless

[1] $[m]$ denotes the set $\{0, \ldots, m-1\}$.

[2] All logarithms in this paper are to the base 2. Note that $\log e \approx 1.443\ldots$

J. Hromkovič et al. (Eds.): SAGA 2007, LNCS 4665, pp. 2–17, 2007.

noted otherwise, we will assume in the following that $n > w \geq (2+\varepsilon)\log n$, and subsume the term $\log w$ in the space bounds in terms $O(\log n)$ and larger.

### 1.1 Space-Optimal, Time-Efficient Constructions

The (information-)theoretical background settled, the question is how close to the bound $n \log e + \log w$ one can get if one insists on constant evaluation time. In the seminal paper [19] Fredman, Komlós, and Szemerédi constructed a dictionary with constant lookup time, which can be used to obtain a MPHF data structure with constant evaluation time and space $O(n \log n)$ bits. Based on [19], Schmidt and Siegel [28] gave a construction for MPHF with constant evaluation time and space $O(n)$ bits (optimal up to a constant factor). Finally, Hagerup and Tholey [20] described a method that in expected linear time constructs a data structure $D_h$ with $n + \log w + o(n + \log w)$ bits, for evaluating a MPHF $h$ in constant time. This is space-optimal up to an additive term. It seems hard, though, to turn the last two constructions into data structures that are space efficient and practically time efficient at the same time for realistic values of $n$.

### 1.2 Practical Solutions

In a different line of development, methods for constructing MPHF were studied that emphasized the evaluation time and simple construction methods over optimality of space. Two different lines (a "graph/hypergraph-based approach" and a method called "hash-and-displace") in principle led to constructions of very simple structures that offered constant evaluation time and a space requirement that was dominated by a table of $\Theta(n)$ elements of $[n] = \{0, \ldots, n-1\}$, which means $\Theta(n \log n)$ bits. Very recently, refinements of these methods were proposed that lead to a space requirement of $O(n \log \log n)$ bits (and constant evaluation time) [11,32]. Only in 2007, Botelho, Pagh, and Ziviani [5] managed to devise a construction for a MPHF that is simple and time-efficient, and gets by with $O(n)$ bits of storage space, with a constant factor that is only a small factor away from the information theory minimum $\log e \approx 1.44$. Crucial steps in this development will be described in some detail in the rest of this paper.

### 1.3 Randomness Assumptions

Given a universe $U$ of keys, a *hash function* is just any function $h: U \to [m]$. Most constructions of MPHF involve several hash functions, which must behave randomly in some way or the other. There are two essentially different ways to approach the issue of the hash functions:

*The "full randomness" assumption*: One assumes that a sequence $h_0, h_1, \ldots$ of hash functions is available, so that evaluating $h_i(x)$ takes constant time, no storage space is needed for these functions, and such that $h_i(x)$, $x \in S$, $i \geq 0$, are fully random values (uniform in $[m]$, independent). The analysis of several MPHF algorithms is based on this assumption (e. g., [8,22,7,4]).
*Randomization*: "Universal hashing" was introduced by Carter and Wegman [6] in 1979. One uses a whole set ("class") $\mathcal{H}$ of hash functions and chooses one such

function from $\mathcal{H}$ at random whenever necessary. Normally, some parameters of a function with a fixed structure are chosen at random. Storing the function means storing the parameters; the analysis is carried out on the basis of the probability space induced by the random choice of the function. Some classical MPFH algorithm use this approach (e. g., [28,25,20]).

Below, we will explain in detail how in the context of the MPHF problem one may quite easily work around the randomness issue by using very simple universal hash classes. To be concrete, we describe two such classes here. We identify $U = \{0,1\}^w$ with $[2^w]$.

**Definition 1.** *A set $\mathcal{H}$ of functions from $U$ to $[m]$ is called 1-universal if for each pair of different $x, y \in U$ and for $h$ chosen at random from $\mathcal{H}$ we have*

$$\Pr(h(x) = h(y)) \leq \frac{1}{m}.$$

There are many constructions of 1-universal classes. One is particularly simple (see [6]): Assume $p$ is a prime number larger than $2^w$, and $m \leq 2^w$. For $a, b \in [p]$ define $h_{a,b}(x) = ((ax+b) \bmod p) \bmod m$, and let $\mathcal{H}_m = \{h_{a,b} \mid a \in [p] - \{0\}, b \in [p]\}$. Choosing/storing a hash function from $\mathcal{H}_m$ amounts to choosing/storing the coefficients $a$ and $b$ (not much more than $2w$ bits).

**Definition 2.** *Let $k \geq 2$. A set $\mathcal{H}$ of functions from $U$ to $[m]$ is called $k$-wise independent if for each sequence $(x_1, \ldots, x_k)$ of different elements of $U$ and for $h$ chosen at random from $\mathcal{H}$ we have that the values $h(x_1), \ldots, h(x_k)$ are fully random in $[m]^k$ and each value $h(x)$ is [approximately] uniformly distributed in $[m]$.*

The simplest way of obtaining a $k$-wise independent class is by using polynomials. Let $p > 2^w$ be a prime number as before, and let $m^{1+\varepsilon} \leq 2^w$ for some $\varepsilon > 0$. The set $\mathcal{H}_m^k$ of all functions of the form

$$h(x) = ((a_{k-1}x^{k-1} + \cdots + a_1 x + a_0) \bmod p) \bmod m, \quad a_{k-1}, \ldots, a_0 \in [p]$$

(polynomials over the field $\mathbf{Z}_p$ of degree smaller than $k$, projected into $[m]$), is $k$-wise independent. Choosing/storing a hash function amounts from this class amounts to choosing/storing the coefficients $(a_{k-1}, \ldots, a_0)$. For details see, e. g., [15,12]. The evaluation time is $\Theta(k)$. For more sophisticated hash function constructions see e. g. [29,14,30].

## 2 Split-and-Share for MPHFs

Let $S \subseteq U$ be fixed, $n = |S|$. For a hash function $h\colon S \to [m]$ and $i \in [m]$ let $S_i = \{x \in S \mid h(x) = i\}$, and let $n_i = |S_i|$. It is a common idea, used many times before in the context of perfect hashing constructions (e. g. in [19,20,10]), to construct separate and disjoint data structures for the "chunks" $S_i$.

The new twist is to "share randomness" among the chunks $S_i$, as follows. (The approach was sketched, for different applications, in [17,16].) In the static

setting, with $S$ given, this works as follows: Choose $h$, and calculate the sets $S_i = \{x \in S \mid h(x) = i\}$ and their sizes $n_i$, repeating if necessary until the sizes are suitable. Then devise one data structure that for each $i$ provides one or several hash functions that behave fully randomly on $S_i$. Each $S_i$ may own some component of this data structure but one essential part (usually a big table of random words) is used ("shared") by all $S_i$'s.

We describe the approach in more detail. First, we "split", and make sure that none of the chunks is too large. The proof of the following lemma is standard.

**Lemma 1.** *If $m \geq 2n^{2/3}$ and $h: U \to [m]$ is chosen at random from a 4-universal class $\mathcal{H} = \mathcal{H}_m^4$, then* $\Pr(\max\{|S_i| \mid 0 \leq i < m\} > \sqrt{n}) \leq \frac{1}{4}$.

*Proof.* The probability that $|S_i| > \sqrt{n}$ is bounded by

$$\Pr\left(\binom{|S_i|}{4} \geq \binom{\sqrt{n}}{4}\right) \leq \frac{\mathrm{E}(\binom{|S_i|}{4})}{\binom{\sqrt{n}}{4}} \leq \frac{\binom{n}{4}/(2n^{2/3})^4}{\binom{\sqrt{n}}{4}} < \frac{1}{8n^{2/3}},$$

for $n$ large enough; hence $\Pr(\exists i: |S_i| \geq \sqrt{n}) \leq 2n^{2/3}/(8n^{2/3}) = \frac{1}{4}$.

Given $S$, we fix $m = 2n^{2/3}$ and repeatedly choose $h$ from $\mathcal{H}_m^4$ until an $h$ with $\max\{|S_i| \mid 0 \leq i < m\} \leq \sqrt{n}$ is found. We fix this function $h$ and call it $h^0$ from here on; thus also the $S_i$ and the $n_i$ are fixed. With $a_i = \sum_{0 \leq j < i} n_i$ we can allocate indices in the interval $[a_i, a_{i+1} - 1]$ as possible hash values for keys in $S_i$.

Once we have found MPHFs $h_i$, one for each $S_i$, we may let

$$h(x) = a_i + h_i(x) \text{ for } i = h^0(x), \tag{1}$$

thus obtaining an MPHF for all of $S$. Below, we will describe several methods for building such a MPHF $h_i$. For this, it is most convenient to have at our disposal one or several hash functions that behave fully randomly (on each $S_i$ separately). To make this concrete, let $K > 1$ be some constant, and let $L = K \log n$. We will argue that when considering $S_i$ we may assume that we have a source of $L$ fully random hash functions $h_1, \ldots, h_L$ from $U$ to $\{0,1\}^k$ for some $k$ we may choose, which can be evalutated in (small) constant time. The data structure that provides the random elements used in these functions will be shared among the different $h_i$.

Let $\mathcal{H}_r$ denote an arbitrary 1-universal class of functions from $U$ to $[r]$.

**Lemma 2.** *Let $r = 2n^{3/4}$. For an arbitrary given $S' \subseteq U$ with $n' = |S'| \leq \sqrt{n}$ we may in expected time $O(|S'|)$ find two hash functions $h_0, h_1$ from $\mathcal{H}_r$ such that for any two tables $T_0[0..r-1]$ and $T_1[0..r-1]$, each containing $r$ random elements from $\{0,1\}^k$, we have that $h'(x) = T_0[h_0(x)] \oplus T_1[h_1(x)]$ defines a function $h': U \to \{0,1\}^k$ that is fully random on $S'$.* ($\oplus$ denotes bitwise XOR.)

*Proof.* Assume $h_0, h_1$ are chosen at random from $\mathcal{H}_r$. We call a pair $h_0, h_1$ *good* if for each $x \in S'$ there is some $i \in \{0,1\}$ such that $h_i(x) \neq h_i(y)$ for all $y \in S' - \{x\}$. For each $x \in S'$, the probability that $\exists y_0 \in S' - \{x\}: h_0(x) = h_0(y_0)$

*and* $\exists y_1 \in S' - \{x\}: h_1(x) = h_1(y_1)$ is smaller than $(\sqrt{n}/r)^2 \leq 1/(4\sqrt{n})$. This implies that the probability that $(h_0, h_1)$ is not good is bounded by $\frac{1}{4}$. We keep choosing $h_1, h_2$ from $\mathcal{H}_r$ until a good pair is found — the expected number of trials is smaller than $\frac{4}{3}$. Checking one pair $h_1, h_2$ takes time $O(|S'|)$ when utilizing an auxiliary array of size $r$. Once a good pair $h_1, h_2$ has been fixed, for a key $x \in S'$ either table position $T_0[h_0(x)]$ or table position $T_1[h_1(x)]$ appears in the calculation of $h(x)$ but of no other key $y \in S'$. Since this entry is fully random, and because $\{0,1\}^k$ with $\oplus$ is a group, $h(x)$ is random and independent of the other hash values $h(y)$, $y \in S' - \{x\}$.

From here, we proceed as follows: For each $i$, $0 \leq i < m$, we choose hash functions $h_0^i, h_1^i$ that are as required in Lemma 2 for $S' = S_i$. The descriptions of these $2m$ hash functions as well as the sizes $n_i$ and the offsets $a_i$ can be stored in (an array that takes) space $O(m) = O(n^{3/4})$ (words of length $O(w)$).

Now we describe the "shared" part of the data structure: Recall that $L = K \log n$. For each $j \in [L]$ we initialize arrays $T_{j,0}[0..r-1]$ and $T_{j,1}[0..r-1]$ with random words from $\{0,1\}^k$. We let

$$h_j^i(x) = T_{j,0}[h_0^i(x)] \oplus T_{j,1}[h_1^i(x)], \text{ for } x \in U,\ 0 \leq j < L,\ 0 \leq i < m.$$

Since $h_0^i, h_1^i$ satisfy the condition in Lemma 2, for each fixed $i$ we have that the values $h_{i,j}(x), x \in S_i, j \in [L]$, are fully random. The overall data structure takes up space $2n^{3/4} \cdot L$ words from $\{0,1\}^k$ plus $O(n^{2/3})$ words of size $\log |U|$, for the description of the $h_0^i, h_1^i$. We will see below that with high probability these hash functions will be sufficient for constructing a MPHF $h_i$ for $S_i$, for all $i \in [m]$. If that construction is not successful, we start all over, with new random entries in the arrays $T_{j,0}$ and $T_{j,1}$ .

From here on we *assume* that we have a fixed set $S'$ of size $n' \leq \sqrt{n}$ and a supply of $L = K \log n$ fully random hash functions $h_0, \ldots, h_{L-1}$ with constant evaluation time and range $\{0,1\}^k$ (identified with $[2^k]$).

**Goal:** Build a MPHF for $S'$ that has constant evaluation time and requires little storage space (beyond the functions $h_0, \ldots, h_{L-1}$). In the rest of the paper we discuss various strategies for achieving this.

## 3 Hash-and-Displace Approach

In this section, we discuss an approach to obtaining a MPHF by splitting $S'$ into buckets, hashing the buckets into the common range $[n']$ and adjusting by offsets.

### 3.1 Pure Hash-and-Displace

Pagh [25] introduced the following approach for constructing a minimal perfect hash function for a set $S'$: Choose hash functions $f: U \to [n']$ and $g: U \to [m']$. The set $[m'] \times [n']$ may be thought of as an array $A$ with entry at $(i,j)$ equal to 1 if $(f(x), g(x)) = 1$ for some $x \in S$, and 0 otherwise. Let $B_i = \{x \in S' \mid g(x) = i\}$,