Hashing, Load Balancing and Multiple Choice

Udi Wieder
VMware Research
udi.wieder@gmail.com

the essence of knowledge
Boston — Delft

Foundations and Trends® in Theoretical Computer Science

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

U. Wieder. *Hashing, Load Balancing and Multiple Choice*. Foundations and Trends® in Theoretical Computer Science, vol. 12, no. 3-4, pp. 275–379, 2016.

This Foundations and Trends® issue was typeset in LaTeX using a class file designed by Neal Parikh. Printed on acid-free paper.

ISBN: 978-1-68083-282-2

Foundations and Trends® in Theoretical Computer Science

Volume 12, Issue 3-4, 2016

Editorial Scope

Topics

Foundations and Trends® in Theoretical Computer Science publishes surveys and tutorials on the foundations of computer science. The scope of the series is broad. Articles in this series focus on mathematical approaches to topics revolving around the theme of efficiency in computing. The list of topics below is meant to illustrate some of the coverage, and is not intended to be an exhaustive list.

- Algorithmic game theory
- Computational algebra
- Computational aspects of combinatorics and graph theory
- Computational aspects of communication
- Computational biology
- Computational complexity
- Computational geometry
- Computational learning
- Computational Models and Complexity
- Computational Number Theory
- Cryptography and information security
- Data structures
- Database theory
- Design and analysis of algorithms
- Distributed computing
- Information retrieval
- Operations research
- Parallel algorithms
- Quantum computation
- Randomness in computation

Information for Librarians

Foundations and Trends® in Theoretical Computer Science, 2016, Volume 12, 4 issues. ISSN paper version 1551-305X. ISSN online version 1551-3068. Also available as a combined paper and online subscription.

Foundations and Trends® in
Theoretical Computer Science
Vol. 12, No. 3-4 (2016) 275–379

DOI: 10.1561/0400000070

now
the essence of knowledge

Hashing, Load Balancing and Multiple Choice

Udi Wieder
VMware Research
udi.wieder@gmail.com

Contents

1 Introduction **2**
1.1 The balls-into-bins model 4
1.2 The Dictionary Data Structure 5

2 Simple Hashing - the One Choice Scheme **6**

3 Multiple Choice Schemes **11**
3.1 The Lightly Loaded Greedy[d] process 12
3.2 The Left[d] Process 17
3.3 Alternative proof techniques 20

4 The Heavily Loaded Case **24**
4.1 General Placement Processes 25
4.2 Back to Greedy[d] 40
4.3 The Power of Majorization 52
4.4 A Lower Bound 60
4.5 Adaptive Schemes 61

5 Dictionaries **62**
5.1 Cuckoo Hashing 63
5.2 Some Interesting Variations 73
5.3 Generalized Cuckoo Hashing and k-Orientability 78

5.4 Linear Probing 84
5.5 Explicit hash functions 89

Acknowledgments **94**

References **95**

Abstract

Many tasks in computer systems could be abstracted as distributing items into buckets, so that the allocation of items across buckets is as balanced as possible, and furthermore, given an item's identifier it is possible to determine quickly to which bucket it was assigned. A canonical example is a dictionary data structure, where 'items' stands for key-value pairs and 'buckets' for memory locations. Another example is a distributed key-value store, where the buckets represent locations in disk or even whole servers. A third example may be a distributed execution engine where items represent processes and buckets compute devices, and so on. A common technique in this domain is the use of a *hash-function* that maps an item into a relatively short fixed length string. The hash function is then used in some way to associate the item to its bucket. The use of a hash function is typically the first step in the solution and additional algorithmic ideas are required to deal with collisions and the imbalance of hash values. In this monograph we survey some of these techniques. We focus on multiple choice schemes where items are placed into buckets via the use of several independent hash functions, and typically an item is placed at the least loaded bucket at the time of placement. We analyze the distributions obtained in detail, and show how these ideas could be used to design basic data structures. With respect to data structures we focus on dictionaries, presenting linear probing, cuckoo hashing and many of their variants.

U. Wieder. *Hashing, Load Balancing and Multiple Choice*. Foundations and Trends® in Theoretical Computer Science, vol. 12, no. 3, pp. 275–378, 2016.
DOI: 10.1561/0400000070.

1

Introduction

'Load Balancing' is a generic name given to a variety of algorithmic problems where a set of items need to be partitioned across buckets, so that the load of each bucket, however defined, is approximately evenly distributed. Phrased in such general terms, the task of load balancing is one of the most fundamental and commonly addressed algorithmic challenges. Typical applications include storage systems where buckets are disks and items files or blocks, data structures where buckets are memory locations and items are keys or distributed execution engines where buckets are servers and items are processes, etc..

This monograph presents some of the basic algorithmic ideas that underpin many of the practical and theoretically interesting approaches for this problem. The most basic building block is a hash function which maps the domain of items into the set of buckets. The hash function is sampled from some family and thus in effect is assumed to be 'random' in some precise way. For instance, if the hash function is sampled uniformly from the set of all possible mappings of items to buckets, then each item is mapped in effect to a uniformly sampled bucket, and the mapping of each item is independent of all other items. In this case the number of items mapped to a given item has the Binomial distribution and bounds on the maximum load could be understood by a fairly standard analysis of the tail of the Binomial distribution (see Section 2).

The first half of this monograph focuses on an algorithmic schema called the *multiple-choice scheme*, named this way because it employs the use of multiple hash functions. On a high level, when there are multiple hash functions each item is mapped to multiple buckets and therefore the algorithm designer has freedom to choose in which of those the item would reside. It turns out that this freedom allows for algorithms which obtain allocations that are much more balanced then that obtained by a single hash function. We will present the main algorithmic ideas and the main mathematical tools that are used for proving bounds on the allocations these algorithms produce. We will see that the analysis is robust to variations in the basic model which in our view explains the effectiveness of these algorithms in practical applications. Our starting point is the simple balls-into-bins model which was essentially presented above but is put forth more formally in Section 1.1. Throughout Sections 2, 3, 4 we examine in detail multiple choice techniques.

The key takeaways a reader should obtain are a familiarity with two powerful proof techniques - the layered induction approach (Section 3) and the potential function based argument (Section 4). These two proof techniques are quite robust and are typically used also for variations over the basic model. A prime example is the Left$[d]$ process, see Section 3.2.

In the second half of the monograph we focus on the dictionary data structure. A dictionary is a fundamental and widely used abstract data structure that supports insertions, deletions and lookups of items. It turns out that efficient implementations of dictionaries borrow substantially from the theory of load balancing algorithms, most notably in a scheme called *cuckoo-hashing* which we present along with many of its variants in Section 5. Finally we discuss the linear probing dictionary, which while not a part of the multiple-choice schema is commonly used and fast in practice.

1.1 The balls-into-bins model

A common framework for reasoning about load balancing processes is that of 'balls' and 'bins' where balls represent the demand (keys, processes, files etc..) and 'bins' represent the supply of resources (table slots, servers, storage units etc..). Throughout this monograph we use the terms buckets and bins as well as items and balls interchangeably.

In this setting we have m balls that are thrown into n bins, typically sequentially according to some allocation rule. The goal is to understand the allocation of balls into bins at the end of the process, usually bounding the load (=number of balls) in the most loaded bin. In this model balls are assigned to bins via one or more *hash functions*. These are functions that map a ball's unique i.d. (typically implicit in the model) to the set of bins, typically numbered $1...n$. Using a hash function to map a bin to a ball, as opposed to simply drawing a bin at random, is useful in the common case where at some subsequent time, a ball's location needs to be recovered from its i.d..

The Random Hashing Assumption Throughout most of the monograph we make the assumption that the hash functions we use are *fully random*. That is, h(ball.id) is a uniformly sampled bin, independent of $h(\cdot)$ for all other balls. Another way of saying it is that the family of functions H from which h is uniformly sampled is the family of all functions from the universe of bin i.d's to the set of bins. Further, we ignore the time it takes to compute h and the space it takes to store it. This assumption allows us to focus on the probabilistic properties of the allocation while ignoring the details of specifying and evaluating an explicit function. In fact, under this assumption, when describing an algorithm, it is sometimes convenient to suppress the existence of a hash function altogether, and just assume that each item 'samples' a bin in a uniform and independent manner. In practice however a specific and explicit hash function has to be implemented, and one has to take into account not only the probabilistic properties of the hash function but also the space required to store it and the time required to compute it. One can quickly observe that a fully random hash function is too expensive to implement in realistic scenarios, as its complexity would dominate the algorithm it serves. A vast body of work is dedicated to removing this assumption and exploring time/space/randomness trade-offs, often for specific applications. The starting point of this line of research is the seminal work of Carter and Wegman [24] on universal hashing. In this monograph we typically stick with the random hashing assumptions, but for further reading see Section 5.5.

1.2 The Dictionary Data Structure

A *dictionary* is a data structure that stores *key,value* pairs and supports the operations of insert(*key,value*), delete(*key*) and lookup(*key*). It is one of the oldest and most widely used data structures, already implemented in the 50's c.f [46, 92]. Numerous implementations exist in essentially all standard libraries. There are many possible ways to implement dictionaries with different algorithmic ideas, and we review some of them in depth in Section 5, but as a primer consider the most basic design called a *simple chained hash table*. The idea is to use a hash function h, that maps the domain of keys to the set $[n]$. An array A of length n is allocated. Ideally we would like the insertion procedure of a key-value pair (k, v) to simply place (k, v) in $A[h(k)]$. This is not attainable since more than one key may be mapped to the same index in the array, a phenomena known as *hash collisions*. In the simple chaining hash table the issue is resolved by having each element of the array be a head pointer of a linked list which connects all the items mapped to that index of the array. Now the insertion procedure places the pair (k, v) in the linked list starting at $A[h(k)]$. Similarly, the procedure lookup(k) searches for the key k in the same linked list.

There may be different ways to perform the actual insertion to the list, but either way the running time of the lookup operation may be as large as the number of items mapped to each index of the array; i.e., to the maximal length of the linked lists. Bounding the length of the lists falls neatly within the balls-into-bins model and is the topic of the next section.

References

[1] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7(4):567–583, December 1986.

[2] Noga Alon, Martin Dietzfelbinger, Peter Bro Miltersen, Erez Petrank, and Gábor Tardos. Linear hash functions. *J. ACM*, 46(5):667–683, September 1999.

[3] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost k-wise independent random variables. *Random Struct. Algorithms*, 3(3):289–304, 1992.

[4] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Addendum to "simple construction of almost k-wise independent random variables". *Random Struct. Algorithms*, 4(1):119–120, 1993.

[5] Noga Alon and Joel Spencer. *The Probabilistic Method*. John Wiley, 1992.

[6] Yuriy Arbitman, Moni Naor, and Gil Segev. De-amortized cuckoo hashing: Provable worst-case performance and experimental results. In *Automata, Languages and Programming, 36th International Colloquium, ICALP*, pages 107–118, 2009.

[7] Yuriy Arbitman, Moni Naor, and Gil Segev. Backyard cuckoo hashing: Constant worst-case operations with a succinct representation. In *51st Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 787–796, 2010.

[8] Martin Aumüller, Martin Dietzfelbinger, and Philipp Woelfel. Explicit and efficient hash families suffice for cuckoo hashing with a stash. *Algorithmica*, 70(3):428–456, 2014.

[9] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. *SIAM J. Comput.*, 29(1):180–200, September 1999.

[10] Tugkan Batu, Petra Berenbrink, and Colin Cooper. Chains-into-bins processes. *J. Discrete Algorithms*, 14:21–28, 2012.

[11] Petra Berenbrink, André Brinkmann, Tom Friedetzky, and Lars Nagel. Balls into non-uniform bins. *J. Parallel Distrib. Comput.*, 74(2):2065–2076, 2014.

[12] Petra Berenbrink, Artur Czumaj, Angelika Steger, and Berthold Vöcking. Balanced allocations: The heavily loaded case. *SIAM J. Comput.*, 35(6):1350–1385, 2006.

[13] Petra Berenbrink, Tom Friedetzky, Zengjian Hu, and Russell A. Martin. On weighted balls-into-bins games. *Theoretical Computuer Science*, 409(3):511–520, December 2008.

[14] Petra Berenbrink, Tom Friedetzky, Peter Kling, Frederik Mallmann-Trenn, Lars Nagel, and Christopher Wastell. Self-stabilizing balls & bins in batches: The power of leaky bins. *CoRR*, abs/1603.02188, 2016.

[15] Petra Berenbrink, Tom Friedetzky, Peter Kling, Frederik Mallmann-Trenn, Lars Nagel, and Christopher Wastell. Self-stabilizing balls & bins in batches: The power of leaky bins. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC*, pages 83–92, 2016.

[16] Petra Berenbrink, Kamyar Khodamoradi, Thomas Sauerwald, and Alexandre Stauffer. Balls-into-bins with nearly optimal load distribution. In *25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA*, pages 326–335, 2013.

[17] Paul Bogdan, Thomas Sauerwald, Alexandre Stauffer, and He Sun. Balls into bins via local search. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 16–34, 2013.

[18] Mark Braverman. Polylogarithmic independence fools ac0 circuits. *J. ACM*, 57(5):28:1–28:10, June 2008.

[19] Karl Bringmann, Thomas Sauerwald, Alexandre Stauffer, and He Sun. Balls into bins via local search: cover time and maximum load. In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 25 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 187–198, 2014.

[20] Andrei Z. Broder and Anna R. Karlin. Multilevel adaptive hashing. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '90, pages 43–53, 1990.

[21] Andrei Z. Broder and Michael Mitzenmacher. Using multiple hash functions to improve IP lookups. In *Proceedings IEEE INFOCOM 2001, The Conference on Computer Communications*, pages 1454–1463, 2001.

[22] John Byers, Jeffrey Considine, and Michael Mitzenmacher. Simple load balancing for distributed hash tables. In *Peer-to-Peer Systems II: Second International Workshop, IPTPS*, pages 80–87, 2003.

[23] Julie Anne Cain, Peter Sanders, and Nicholas C. Wormald. The random graph threshold for k-orientiability and a fast algorithm for optimal multiple-choice allocation. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 469–476, 2007.

[24] J.Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143 – 154, 1979.

[25] Cassandra. Apache. http://cassandra.apache.org/.

[26] L. Elisa Celis, Omer Reingold, Gil Segev, and Udi Wieder. Balls and bins: Smaller hash families and faster evaluation. *SIAM J. Comput.*, 42(3):1030–1050, 2013.

[27] X. Chen. Derandomized Balanced Allocation. *ArXiv e-prints*, February 2017.

[28] Tobias Christiani, Rasmus Pagh, and Mikkel Thorup. From independence to expansion and back again. In *Proceedings of the 47th Annual ACM on Symposium on Theory of Computing, STOC*, pages 813–820, 2015.

[29] Artur Czumaj and Volker Stemann. Randomized allocation processes. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 194–203, 1997.

[30] Søren Dahlgaard, Mathias Bæk Tejs Knudsen, Eva Rotenberg, and Mikkel Thorup. Hashing for statistics over k-partitions. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1292–1310, 2015.

[31] Søren Dahlgaard, Mathias Bæk Tejs Knudsen, Eva Rotenberg, and Mikkel Thorup. The power of two choices with simple tabulation. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1631–1642, 2016.

[32] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: Amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, 41(6):205–220, October 2007.

[33] Luc Devroye and Pat Morin. Cuckoo hashing: Further analysis. *Information Processing Letters*, 86:215–219, 2003.

[34] Martin Dietzfelbinger. Universal hashing and k-wise independent random variables via integer arithmetic without primes. In *STACS 96, 13th Annual Symposium on Theoretical Aspects of Computer Science, Grenoble, France, February 22-24, 1996, Proceedings*, pages 569–580, 1996.

[35] Martin Dietzfelbinger, Joseph Gil, Yossi Matias, and Nicholas Pippenger. Polynomial hash functions are reliable. In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, ICALP '92, pages 235–246. Springer-Verlag, London, UK, 1992.

[36] Martin Dietzfelbinger, Andreas Goerdt, Michael Mitzenmacher, Andrea Montanari, Rasmus Pagh, and Michael Rink. Tight thresholds for cuckoo hashing via XORSAT. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*, pages 213–225, 2010.

[37] Martin Dietzfelbinger and Friedhelm Meyer auf der Heide. A new universal class of hash functions and dynamic hashing in real time. In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, ICALP '90, pages 6–19. Springer-Verlag, London, UK, 1990.

[38] Martin Dietzfelbinger, Anna R. Karlin, Kurt Mehlhorn, Friedhelm Meyer auf der Heide, Hans Rohnert, and Robert Endre Tarjan. Dynamic perfect hashing: Upper and lower bounds. *SIAM J. Comput.*, 23(4):738–761, 1994.

[39] Martin Dietzfelbinger and Michael Rink. Applications of a splitting trick. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I*, ICALP '09, pages 354–365. Springer-Verlag, Berlin, Heidelberg, 2009.

[40] Martin Dietzfelbinger and Ulf Schellbach. On risks of using cuckoo hashing with simple universal hash classes. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 795–804, Philadelphia, PA, USA, 2009.

[41] Martin Dietzfelbinger and Christoph Weidling. Balanced allocation and dictionaries with tightly packed constant size bins. *Theoretical Computer Science*, 380(1-2):47–68, 2007.

[42] Martin Dietzfelbinger and Philipp Woelfel. Almost random graphs with simple hash functions. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 629–638, 2003.

[43] Gregory Dresden and Du Zhaohui. A simplified binet formula for k-generalized fibonacci numbers. *Journal of Integer Sequences*, 17, 2014.

[44] Michael Drmota and Reinhard Kutzelnigg. A precise analysis of cuckoo hashing. *ACM Trans. Algorithms*, 8(2):11, 2012.

[45] Devdatt Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

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

[47] Daniel Fernholz and Vijaya Ramachandran. The k-orientability thresholds for $G_{n,\ p}$. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 459–468, 2007.

[48] Dimitris Fotakis, Rasmus Pagh, Peter Sanders, and Paul G. Spirakis. Space efficient hash tables with worst case constant access time. *Theory Comput. Syst.*, 38(2):229–248, 2005.

[49] Nikolaos Fountoulakis, Megha Khosla, and Konstantinos Panagiotou. The multiple-orientability thresholds for random hypergraphs. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1222–1236, 2011.

[50] Nikolaos Fountoulakis and Konstantinos Panagiotou. Sharp load thresholds for cuckoo hashing. *Random Struct. Algorithms*, 41(3):306–333, 2012.

[51] Nikolaos Fountoulakis, Konstantinos Panagiotou, and Angelika Steger. On the insertion time of cuckoo hashing. *SIAM J. Comput.*, 42(6):2156–2181, 2013.

[52] Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with 0(1) worst case access time. *J. ACM*, 31(3):538–544, June 1984.

[53] Alan Frieze, Páll Melsted, and Michael Mitzenmacher. An analysis of random-walk cuckoo hashing. *SIAM J. Comput.*, 40(2):291–308, March 2011.

[54] Alan M. Frieze and Tony Johansson. On the insertion time of random walk cuckoo hashing. *CoRR*, abs/1602.04652, 2016.

[55] Alan M. Frieze and Páll Melsted. Maximum matchings in random bipartite graphs and the space utilization of cuckoo hash tables. *Random Struct. Algorithms*, 41(3):334–364, 2012.

[56] Pu Gao and Nicholas C. Wormald. Orientability thresholds for random hypergraphs. *Combinatorics, Probability & Computing*, 24(5):774–824, 2015.

[57] P. Brighten Godfrey. Balls and bins with structure: Balanced allocations on hypergraphs. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 511–517, Philadelphia, PA, USA, 2008.

[58] Michael Goodrich, Evgenios Kornaropoulos, Michael Mitzenmacher, and Roberto Tamassia. More practical and secure history-independent hash tables. In *European symposium on Research in Computer Security*, 2016.

[59] David Karger, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, STOC '97, pages 654–663, New York, NY, USA, 1997.

[60] Richard M. Karp, Michael Luby, and Friedhelm Meyer auf der Heide. Efficient pram simulation on a distributed memory machine. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, STOC '92, pages 318–326, New York, NY, USA, 1992.

[61] Krishnaram Kenthapadi and Rina Panigrahy. Balanced allocation on graphs. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 434–443, 2006.

[62] Megha Khosla. Balls into bins made faster. In *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 601–612, 2013.

[63] Adam Kirsch and Michael Mitzenmacher. Using a queue to de-amortized cuckoo hashing in hardware. In *the 45th Annual Allerton Conference on Communication, Control, and Computing*, 2007.

[64] Adam Kirsch, Michael Mitzenmacher, and Udi Wieder. More robust hashing: Cuckoo hashing with a stash. *SIAM J. Comput.*, 39(4):1543–1561, December 2009.

[65] Toryn Qwyllyn Klassen and Philipp Woelfel. Independence of tabulation-based hash classes. In *LATIN 2012: Theoretical Informatics - 10th Latin American Symposium, Arequipa, Peru, April 16-20, 2012. Proceedings*, pages 506–517, 2012.

[66] Donald Knuth. Notes on open addressing. 1963.

[67] Reinhard Kutzelnigg. A further analysis of cuckoo hashing with a stash and random graphs of excess r. *Discrete Mathematics & Theoretical Computer Science*, 12(3):81–102, 2010.

[68] Eric Lehman and Rina Panigrahy. 3.5-way cuckoo hashing for the price of 2-and-a-bit. In Amos Fiat and Peter Sanders, editors, *ESA*, volume 5757 of *Lecture Notes in Computer Science*, pages 671–681. Springer, 2009.

[69] Marc Lelarge. A new approach to the orientation of random hypergraphs. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 251–264, 2012.

[70] Daniel Lemire. The universality of iterated hashing over variable-length strings. *Discrete Applied Mathematics*, 160(4-5):604–617, 2012.

[71] Christoph Lenzen and Roger Wattenhofer. Tight bounds for parallel randomized load balancing: Extended abstract. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, STOC '11, pages 11–20, New York, NY, USA, 2011.

[72] Chi-Jen Lu. Improved pseudorandom generators for combinatorial rectangles. *Combinatorica*, 22(3):417–434, 2002.

[73] Raghu Meka, Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Fast pseudorandomness for independence and load balancing - (extended abstract). In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 859–870, 2014.

[74] Daniele Micciancio. Oblivious data structures: Applications to cryptography. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, STOC '97, pages 456–464, New York, NY, USA, 1997.

[75] Michael Mitzenmacher. *The Power of Two Choices in Randomized Load Balancing*. PhD thesis, University of California at Berkeley, 1996.

[76] Michael Mitzenmacher. Balanced allocations and double hashing. In *26th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '14, Prague, Czech Republic - June 23 - 25, 2014*, pages 331–342, 2014.

[77] Michael Mitzenmacher, Andréa W. Richa, and Ramesh Sitaraman. The power of two random choices: A survey of techniques and results. In *in Handbook of Randomized Computing*, pages 255–312. Kluwer, 2000.

[78] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[79] Michael Mitzenmacher and Salil Vadhan. Why simple hash functions work: Exploiting the entropy in a data stream. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 746–755, Philadelphia, PA, USA, 2008.

[80] Michael Mitzenmacher and Berhold Vöcking. The asymptotics of selecting the shortest of two, improved. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control, and Computing*, pages 326–327, 1999.

[81] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[82] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

[83] Moni Naor, Gil Segev, and Udi Wieder. History-independent cuckoo hashing. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, pages 631–642, 2008.

[84] Moni Naor and Vanessa Teague. Anti-persistence: History independent data structures. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, STOC '01, pages 492–501, New York, NY, USA, 2001.

[85] Moni Naor and Udi Wieder. Novel architectures for p2p applications: The continuous-discrete approach. In *Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '03, pages 50–59, New York, NY, USA, 2003.

[86] Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, February 1996.

[87] Anna Pagh and Rasmus Pagh. Uniform hashing in constant time and optimal space. *SIAM J. Comput.*, 38(1):85–96, 2008.

[88] Anna Pagh, Rasmus Pagh, and Milan Ruzic. Linear probing with 5-wise independence. *SIAM Review*, 53(3):547–558, 2011.

[89] Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *J. Algorithms*, 51(2):122–144, May 2004.

[90] Rina Panigrahy. Efficient hashing with lookups in two memory accesses. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 830–839, Philadelphia, PA, USA, 2005.

[91] Yuval Peres, Kunal Talwar, and Udi Wieder. Graphical balanced allocations and the (1 + beta)-choice process. *Random Structures and Algorithms*, 2014.

[92] W. W. Peterson. Addressing for random-access storage. *IBM J. Res. Dev.*, 1(2):130–146, April 1957.

[93] Mihai Pătraşcu and Mikkel Thorup. The power of simple tabulation hashing. *J. ACM*, 59(3):14, 2012.

[94] Mihai Pătraşcu and Mikkel Thorup. Twisted tabulation hashing. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pages 209–228, Philadelphia, PA, USA, 2013.

[95] Mihai Pătraşcu and Mikkel Thorup. On the k-independence required by linear probing and minwise independence. *ACM Trans. Algorithms*, 12(1):8:1–8:27, November 2015.

[96] Martin Raab and Angelika Steger. Balls into bins - a simple and tight analysis. In Michael Luby, Jose D.P. Rolim, and Maria Serna, editors, *Randomization and Approximation Techniques in Computer Science*, volume 1518 of *Lecture Notes in Computer Science*, pages 159–170. Springer Berlin Heidelberg, 1998.

[97] Omer Reingold, Ron D. Rothblum, and Udi Wieder. Pseudorandom graphs in data structures. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 943–954, 2014.

[98] A. Wayne Roberts and Dale E. Varberg. *Convex Functions*. Academic Press Inc., New York, NY, USA, 1st edition, 1973.

[99] Peter Sanders, Sebastian Egner, and Jan Korst. Fast concurrent access to parallel disks. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, pages 849–858, Philadelphia, PA, USA, 2000.

[100] Alan Siegel. On universal classes of extremely random constant-time hash functions. *SIAM J. Comput.*, 33(3):505–543, 2004.

[101] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Comput. Commun. Rev.*, 31(4):149–160, August 2001.

[102] Kunal Talwar and Udi Wieder. Balanced allocations: the weighted case. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 256–265, 2007.

[103] Kunal Talwar and Udi Wieder. Balanced allocations: A simple proof for the heavily loaded case. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming*, volume 8572 of *Lecture Notes in Computer Science*, pages 979–990. Springer Berlin Heidelberg, 2014.

[104] Mikkel Thorup. String hashing for linear probing. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 655–664, 2009.

[105] Mikkel Thorup and Yin Zhang. Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. *SIAM J. Comput.*, 41(2):293–331, 2012.

[106] Berthold Vöcking. How asymmetry helps load balancing. *J. ACM*, 50(4):568–589, July 2003.

[107] Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci.*, 22(3):265–279, 1981.

[108] Udi Wieder. Balanced allocations with heterogenous bins. In *Proceedings of the 19th Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '07, pages 188–193, New York, NY, USA, 2007.

[109] Philipp Woelfel. Asymmetric balanced allocation with simple hash functions. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 424–433, Philadelphia, PA, USA, 2006.