I N F O R M A T I K

Almost Random Graphs with Simple Hash Functions

Martin Dietzfelbinger
and
Philipp Woelfel

MPI–I–2003–1–005 March 2003

FORSCHUNGSBERICHT RESEARCH REPORT

M A X - P L A N C K - I N S T I T U T
F Ü R
I N F O R M A T I K

Stuhlsatzenhausweg 85 66123 Saarbrücken Germany

Authors' Addresses

Martin Dietzfelbinger
Fakultät für Informatik und Automatisierung
Technische Universität Ilmenau
98684 Ilmenau, Germany
email: `martin.dietzfelbinger@tu-ilmenau.de`

Philipp Woelfel
Fachbereich Informatik, Lehrstuhl 2
Universität Dortmund
44221 Dortmund, Germany
email: `philipp.woelfel@cs.uni-dortmund.de`

Abstract

We describe a simple randomized construction for generating pairs of hash functions h_1, h_2 from a universe U to ranges $V = [m] = \{0, 1, \ldots, m-1\}$ and $W = [m]$ so that for every key set $S \subseteq U$ with $n = |S| \le m/(1+\epsilon)$ the (random) bipartite (multi)graph with node set $V \uplus W$ and edge set $\{(h_1(x), h_2(x)) \mid x \in S\}$ exhibits a structure that is essentially random. The construction combines d-wise independent classes for d a relatively small constant with the well-known technique of random offsets. While keeping the space needed to store the description of h_1 and h_2 at $O(n^\zeta)$, for $\zeta < 1$ fixed arbitrarily, we obtain a much smaller (constant) evaluation time than previous constructions of this kind, which involved Siegel's high-performance hash classes. The main new technique is the combined analysis of the graph structure and the inner structure of the hash functions, as well as a new way of looking at the cycle structure of random (multi)graphs. The construction may be applied to improve on Pagh and Rodler's "cuckoo hashing" (2001), to obtain a simpler and faster alternative to a recent construction of Östlin and Pagh (2002/03) for simulating uniform hashing on a key set S, and to the simulation of shared memory on distributed memory machines. We also describe a novel way of implementing (approximate) d-wise independent hashing without using polynomials.

Keywords

random graphs, hash function, cuckoo hashing, uniform hashing

1 Introduction and Overview

1.1 Construction of almost random graphs

This paper is concerned with the study of random graphs[1] that are generated by pairs of hash functions. We consider the following scenario, known from various contexts. Assume a set S of n keys from a universe U and some range $[m] = \{0, 1, \ldots, m-1\}$ are given. Choose two functions h_1 and h_2 from U to $[m]$ at random, according to some distribution. Often, the structure of the (random) graph determined by the edge (multi)set $\{(h_1(x), h_2(x)) \mid x \in S\}$ is essential in the analysis of some algorithm [9, 11, 13, 14, 15, 19]. Slightly different situations result according as one assumes the node set of the graph is $[m]$ or one considers a bipartite graph with node set $V \uplus W$, where V and W are disjoint copies of $[m]$. In this paper, we work with the second option, and assume that $m \geq (1+\epsilon)n$ for some constant $\epsilon > 0$. The bipartite graph determined by S, h_1, and h_2 is denoted by $G(S, h_1, h_2)$.

If the edges $(h_1(x), h_2(x))$, $x \in S$, are assumed to be fully random in $V \times W$, the properties of $G(S, h_1, h_2)$ can be studied using known tools from the theory of random graphs [1]. On the other hand, in the context of randomized algorithms one usually assumes some kind of limited randomness that can be generated by choosing h_1 and h_2 from (universal) hash classes comprising functions with a moderate representation size and small evaluation time.

As a technical basis for the further discussion in this introduction, we define the concept of d-wise independent hash classes and state the properties of Siegel's hash classes.

d-wise independent hash classes. The idea of choosing a random hash function from a *hash class* (i. e., a family of hash functions), instead of using fixed hash functions, dates back to 1979, when Carter and Wegman introduced "*universal hash classes*" [2]. For $d \geq 2$, a universe U and a range size m, a family $\mathcal{H}$ of functions mapping U to $[m]$ is called a *d-wise independent* class (of hash functions) if for h chosen randomly from $\mathcal{H}$ the values $h(x_1), \ldots, h(x_d)$ on d distinct keys $x_1, \ldots, x_d$ are uniformly and independently distributed over $[m]$. For constant $d \geq 2$ (and m a prime power) such classes may easily be constructed. For $d = 2$, many constructions are known, for $d \geq 3$ most standard constructions involve polynomials of degree $< d$ (e. g., [5]). Alternatively, the following construction may be used.

Siegel's high-performance hash classes. Siegel (in the technical report version [23] of [22]) gave a construction to the following effect.

Fact 1. *Let $0 < \mu < 1$ and $k \geq 1$ with $\mu k < 1$ be given. Then if $\zeta < 1$ and $d \geq 1$ satisfy $\zeta \geq \frac{2k}{d} + \frac{1+\log d+\mu\log n}{\zeta\log n} \cdot k$ (for n large enough), then there is a way of randomly choosing a function $h\colon [n^k] \to [n]$ such that the following holds: the description of h comprises $O(n^\zeta)$ words in $[n]$; h can be evaluated by XOR-ing together $d^{k/\zeta}$ $k\log n$-bit words; the class formed by all these h's is n^μ-wise independent.*

This theorem is based on the fact that there are bipartite graphs on the node set $[n^k] \uplus [n]$ with certain expansion properties; such graphs are proved to exist by the probabilistic method. The fact that the description of h must contain a compressed representation of such a graph (so that the space limit $O(n^\zeta)$ is respected) makes the evaluation time quite

[1] Throughout this paper, the word ***graph*** is used in the sense of ***multigraph***, i. e., there may be multiple edges. Accordingly, when we talk about sets of edges, we mean multisets.

large, although still constant. Note that this construction does not use polynomials. For other hash classes with strong randomness properties see [7, 11].

Now we resume the discussion of our basic construction. Up to now, in applications of the graphs $G(S, h_1, h_2)$ mentioned above one had to assume for the analysis that h_1, h_2 are chosen from Siegel's class or related high-performance hash classes, causing a large (constant) evaluation time. In this paper we exhibit a simple construction for functions h_1, h_2 with constant evaluation time, where the constant is much smaller than in Siegel's construction, with the following properties: (i) Both can be evaluated as fast as a polynomial with a (small) constant degree; storing the functions takes space $O(n^\zeta)$, for $\zeta < 1$ an arbitrary constant. (ii) There is a (polynomially in n^{-1}) small probability that a "bad" event happens; outside of the bad event the hash values inside each connected component of $G(S, h_1, h_2)$ behave in a fully random way.

Usually, in applications of the (h_1, h_2)-construction, there is no interaction between parts of the structure that correspond to different connected components. Thus, for the analysis of an application for which this is so we may assume that the graph $G(S, h_1, h_2)$ is fully random. Technically, we design a class of *pairs* of hash functions, by just slightly extending constructions known from [7], and prove some new basic facts about the probability space that governs the structure of the resulting graph. A new way of analyzing the structure of this graph, especially its cycle structure, in combination with the inner structure of the underlying hash functions, is instrumental.

The usefulness of the new result is illustrated by three applications, as described in the following.

1.2 Cuckoo hashing

Cuckoo hashing was introduced by Pagh and Rodler [19] as an algorithm for maintaining a dynamic dictionary with constant lookup time in the worst case. The data structure consists of two tables T_1 and T_2 of size m. Given two hash functions h_1 and h_2 from U to $[m]$, one maintains the invariant that a key x presently stored in the data structure occupies either cell $T_1[h_1(x)]$ or $T_2[h_2(x)]$ (but not both). Given this invariant and the property that h_1 and h_2 may be evaluated in constant time, procedures for lookups and deletions that run in worst case constant time are obvious. Pagh and Rodler describe a simple procedure for inserting a new key x. If cell $T_1[h_1(x)]$ is empty, then x is placed there; if this cell is occupied by a key x_1 (which necessarily satisfies $h_1(x_1) = h_1(x)$), then x is put in cell $T_1[h_1(x)]$ anyway, and x_1 becomes "nestless". Then, one puts x_1 into cell $T_2[h_2(x_1)]$ in the same way, which may leave another key x_2 (with $h_2(x_2) = h_2(x_1)$) "nestless". In this case, x_2 is placed in cell $T_1[h_1(x_2)]$, and one continues until the key that is currently nestless can be placed in an empty cell, or until L replacing steps have been made, for some bound L chosen beforehand. In the latter case the data structure is built anew from scratch with new functions h_1 and h_2, by newly inserting all keys currently stored in the data structure, recursively using the same insertion procedure for each key.

For discussing the time and space requirements, we consider a simple scenario (which may be used in a standard way to obtain a fully dynamic dictionary). We consider a certain period in which the dictionary is in operation (a "phase"). The set of all keys ever stored in the dictionary during the phase is called S, its cardinality is n. The phase comprises arbitrarily many lookups and at most ρn update operations, for a constant ρ; the table size satisfies $m \geq (1+\epsilon)n$, for $\epsilon > 0$ a constant. (Due to insertions and deletions, it is usually not the case that all keys from S are in the dictionary at the same time.) Pagh and Rodler show that if the hash functions h_1 and h_2 are chosen independently from a $c \log n$-wise independent class, for a suitable constant $c > 0$, and if $L = \Theta(\log n)$ is chosen appropriately, then the amortized expected time for inserting x is constant. The time analysis is based on properties of the random graph $G(S, h_1, h_2)$. For details see [19]. The analysis is valid in particular if h_1 and h_2 are chosen independently from a suitable class of hash functions built according to Siegel's construction [22]. Since these functions exhibit quite a large evaluation time, Pagh and Rodler asked for simpler hash classes to be used instead.

If we employ our new hash function pairs in cuckoo hashing, we obtain a much more practical version of this elegant dynamic dictionary with constant lookup time, without appealing to Siegel's functions. This means that the evaluation time for the hash functions involved is much smaller (in terms of constant factors) than in [19]. From experiments in [19] that used hash classes with simpler functions (but for which it is not clear whether the analysis carries through) one already knew that in practice cuckoo hashing is superior to dynamic perfect hashing [8]; our new result shows that cuckoo hashing even in a version that can be theoretically analyzed beats dynamic perfect hashing as far as space requirements go and is at least competitive with respect to evaluation time.

In a recent development, Fotakis *et al.* [10] brought down the space requirement in cuckoo hashing to $(1+\epsilon)n$ by allowing a fixed number $d = d(\epsilon) \geq 2$ of hash functions instead of just two. However, in the analysis it was assumed that the hash functions are fully random, and it is an open problem whether Siegel's functions are sufficient for the analysis to go through or if even our constructions can be adapted to that situation.

1.3 Simulating uniform hashing without Siegel's functions

By analyzing the cycle structure of the graphs $G(S, h_1, h_2)$ generated by functions h_1, h_2 from Siegel's class, Östlin and Pagh [15, 16], in an astonishing development, described a construction to the following effect. Let U be a finite universe, let m be some range size, and let $n \leq |U|$. There is a randomized procedure to choose a function h from U to $[m]$ so that the following holds:

(i) The description of h comprises $O(n^\zeta)$ words from $[n]$, for some $\zeta < 1$, plus $8n + O(n^\zeta)$ words from $[m]$;

(ii) evaluating h amounts to evaluating three functions from Siegel's class, hence the evaluation time of h is constant;

(iii) for every given set $S \subseteq U$ of size n there is an event B_S of probability $O(n^{-c})$ (c an arbitrary given constant), so that if B_S does not occur, the values $h(x)$, $x \in S$, are independent and uniformly distributed in $[m]$.

That it is possible to simulate full randomness in hash functions in linear space and with constant evaluation time (without very complex and space-consuming real-time dictionaries like in [7]) had eluded researchers for many years, even after Siegel's construction had appeared. As explained in [15, 16], Östlin and Pagh's result implies that the idealizing uniformity assumption in the analysis of classical hashing schemes like linear probing or double hashing can be realized by a randomized algorithm, at the cost of $O(n)$ extra space. Schmidt and Siegel [20, 21] proposed an analysis of double hashing and linear probing based on the weaker $O(\log n)$-wise independence, but the analysis in particular of double hashing becomes considerably more complicated than with full independence [12].

In the present paper, we give a simple construction based on the random graphs $G(S, h_1, h_2)$ mentioned above, and avoiding the use of Siegel's functions. In an analysis that takes a different view on the cycle structure of these graphs, we show that the resulting functions have essentially the same randomness properties as those from [15, 16], but have a moderate (constant) evaluation time and smaller, though still linear, space requirements.

1.4 Shared memory simulations

Many algorithms for distributing shared memory to the memory modules of a distributed memory machine (DMM) [11, 14] redundantly store the contents of cell x in several memory modules given by hash functions $h_1(x), \ldots, h_a(x)$. In the simplest case, two hash functions h_1 and h_2 are used. (E. g., Process_3 in [11] is of this type.) The analysis of such an algorithm is based on the properties of the random graph $G(S, h_1, h_2)$ (where S is the set of shared memory cells accessed in one step). Up to now, Siegel's functions had to be used as basis for analyzing this approach; the new construction makes it possible to get by with hash functions with much smaller evaluation time. We expect that our approach can be extended to more complicated analysis methods like the witness tree method from [14] (which involves more than 2 hash functions), as long as the edge density of the involved graphs is not larger than $1/(2(1+\epsilon))$.

1.5 d-wise independence without polynomials

The concept of d-wise independent hash classes (or approximately d-wise independent classes) is well-studied in the area of universal hashing. Most known constructions of d-wise independent classes for $d > 2$ involve polynomials of degree up to $d-1$ over some algebraic structures like fields or rings [2, 4, 5, 24]. The most notable exception is Siegel's class from [23] with functions of a relatively complex structure and a relatively large evaluation time (in particular for small d). Complementing the random graph construction, which uses d-wise independent classes as an ingredient, we give a new construction of approximately d-wise independent classes that do not involve polynomials in the key x, but rather use x only in a linear way. This results in the construction of an approximately d-independent class whose functions can be evaluated by one multiplication of integers of $O(d \log |U|)$ bits, some bit operations, and some table lookups. The drawback is that for range $[m]$ space m^ζ is needed, for some constant $\zeta < 1$. In combination with the results from the first part of the paper we obtain that almost random graphs can be generated by random experiments involving only one integer multiplication plus some table lookups, and that for cuckoo hashing and shared memory simulation such hash functions are sufficient. Uniform hashing can be simulated by functions that involve two multiplications of $O(\log |U|)$-bit numbers.

Structure of this paper

In Section 2.1, the definitions of our hash functions are given and the main theorem is formulated. In Section 3 some graph theoretical facts and the main theorem are proved. In Section 4 the application of our results to cuckoo hashing and the simulation of uniform hashing is described in detail; the application to shared memory simulation is discussed briefly. Finally, Section 5 describes the novel construction of approximately d-wise independent hash classes.

2 Almost random graphs

2.1 Hash classes

Throughout this paper (excepting for Section 5) we assume that for any given U and m we have a d-wise independent hash class $\mathcal{H}_m^d$ of functions $U \to [m]$ at our disposal. Many constructions of such classes are known, with functions that can be evaluated in constant time (on a RAM), and which have cardinality $O(|U|^d)$ [4, 5, 24]. To be precise, d-wise independent hash classes of reasonable size are known to exist only if m is a prime power. For other values of m there are hash classes which are approximately d-wise independent in the sense that the probability that the random vector $(h(x_1), \ldots, h(x_d))$ takes a certain value $(y_1, \ldots, y_d) \in [m]^d$ deviates from $1/m^d$ by at most ϵ/m^d, for some small $\epsilon > 0$. It is no problem to accommodate the analysis to follow to hash classes that are only approximately d-wise independent.

For the whole paper, it is convenient to assume that if $S \subseteq U$ is the key set of interest, and $|S| = n$, then $|U| = n^{O(1)}$. If this is not the case, the well-known technique of "collapsing the universe" is used. We use a function h from U to some auxiliary universe $U' = [n^k]$ to map S to $S' = h(S) \subseteq [n^k]$. If h is chosen at random from some 2-independent class, say, the probability that h is one-to-one on S is $1 - O(1/n^{k-2})$, which is sufficient for all our purposes if the constant k is chosen large enough.

We define now a class of hash function pairs which is the basis for our later constructions of almost random bipartite graphs.

Definition 1. *Let $d \geq 2$ and $r, m \in \mathbb{N}$. For $f \in \mathcal{H}_m^d$, $g \in \mathcal{H}_r^d$, and $z = (z_0, \ldots, z_{r-1}) \in [m]^r$ the hash function $h_{f,g,z} : U \to [m]$ is defined by $x \mapsto \big(f(x) + z_{g(x)}\big) \bmod m$. The hash class $\mathcal{R}_{r,m}^d$ is the family of all functions $h_{f,g,z}$. The family $\hat{\mathcal{R}}_{r,m}^d$ consists of all hash function pairs (h_1, h_2), where $h_i = h_{f_i,g,z^{(i)}}$ with $f_1, f_2 \in \mathcal{H}_m^d$, $g \in \mathcal{H}_r^d$, and $z^{(1)}, z^{(2)} \in [m]^r$.*

Note that we have not used $\mathcal{R}_{r,m}^d \times \mathcal{R}_{r,m}^d$ in order to define pairs of hash functions, but that the functions h_1, h_2 of a pair from $\hat{\mathcal{R}}_{r,m}^d$ share the same g-function. It will turn out that this "dependence" between the two functions is helpful for the analysis.

The idea of using classes of hash functions $h_{f,g,z}$ is due to Dietzfelbinger and Meyer auf der Heide [7], who have thoroughly analyzed a hash class very similar to $\mathcal{R}_{r,m}^d$. Modifications of such hash classes appeared later in different works [11, 17]. The analysis of the behaviour of function pairs from $\hat{\mathcal{R}}_{r,m}^d$ on a given key set S is fundamentally different from previous methods applied to pairs of functions from $\mathcal{R}_{r,m}^d$ in that we consider the interplay between the inner structure of these functions and the structure of the graph $G(S, h_1, h_2)$. Previous analyses only relied on the strong randomness properties of the single functions from $\mathcal{R}^d_{r,m}$ or from Siegel's class.

2.2 Generating almost random graphs

As mentioned before, we will study the following graphs generated by hash function pairs.

Definition 2. *Let h_1, h_2 be hash functions $U \to [m]$ and V and W be disjoint copies of $[m]$. Then each set $S \subseteq U$ defines a bipartite graph $G = G(S, h_1, h_2) = (V \uplus W, E)$, where $E = E(S, h_1, h_2) = \{(h_1(x), h_2(x)) \mid x \in S\}$ (a multiset).*

Our main result indicates that the distribution of graphs $G(S, h_1, h_2)$, if the pair (h_1, h_2) is chosen randomly from $\hat{\mathcal{R}}^d_{r,m}$, is in several aspects similar to that of perfectly random bipartite graphs with $|S|$ edges.

In order to state the main result precisely, we need some graph theoretical definitions. (See, e.g., [3], for basic notions in graph theory.) Let $G = (V, E)$ be a connected graph. We identify each simple cycle and each simple path of G with the set of its edges. The *length* of a path or a cycle is the number of edges it contains. Note that since we are considering graphs with multiple edges there may be simple cycles of length 2 (but no loops, since all our graphs are bipartite). We define an important property of (connected) graphs.

Definition 3. *Let $G = (V, E)$ be a connected graph. Let $T \subseteq E$ be an arbitrary spanning tree of G. The* cyclomatic number *of G is $|E| - |T|$, the number of edges of G not in T.*

Since a spanning tree of a connected n-node graph has $n - 1$ edges, the following is obvious:

Lemma 1. *The cyclomatic number of a connected graph with n vertices and m edges is $m - n + 1$.*

Note that each edge $e \in E - T$ together with T determines a unique simple cycle in G (the *fundamental cycle* of e w.r.t. T). It is well known that every cycle in G may be obtained from the fundamental cycles by repeatedly applying the XOR operation on edge sets, see [3].

We are especially interested in those graphs $G = G(S, h_1, h_2)$ for which the cyclomatic number of each connected component is bounded by a constant. In Section 4.2 we will show how any such graph leads to a hash class which is uniform on S. Moreover, for cuckoo hashing to be successful on S a function pair (h_1, h_2) is needed so that the cyclomatic number of every connected component of $G(S, h_1, h_2)$ is bounded by 1, i. e., each connected component contains at most one cycle. Another desirable property of the random distribution of G is that the expected length of the longest simple path starting at an arbitrary vertex is bounded by a constant, because for cuckoo hashing, the length of the longest simple path starting at $h_1(x)$ is an upper bound on the insertion time of x.

A bipartite graph with $(1 + \epsilon)n$ nodes on each side whose $n = |S|$ edges are placed completely at random has these properties with high probability. Our main result, to be formulated next, says that these properties are also obtained with high probability if (h_1, h_2) are chosen randomly from $\hat{\mathcal{R}}^d_{r,m}$. For each set S, our analysis is only valid for a subset $R(S)$ of "good" hash function pairs from $\hat{\mathcal{R}}^d_{r,m}$, but this causes no problem in applications because a randomly chosen pair is good with high probability. In the proof, the hash function pairs that are not "good" will be identified.

Theorem 1. *Let $\epsilon > 0$ and $\ell \geq 1$ be fixed and let $m = m(n) = (1+\epsilon)n$. For any set $S \subseteq U$, $|S| = n$, there exists a set $R(S)$ of "good" hash function pairs in $\hat{\mathcal{R}}^{2\ell}_{r,m}$ such that for randomly chosen $(h_1, h_2) \in \hat{\mathcal{R}}^{2\ell}_{r,m}$ the following holds.*

(a) $\mathbf{Prob}\big((h_1, h_2) \in R(S)\big) \geq 1 - n/r^{\ell}$.

(b) *For every constant $q \geq 2$ the probability that $(h_1, h_2) \in R(S)$ and that there is a connected subgraph of $G(S, h_1, h_2)$ whose cyclomatic number is at least q is $O\left(n^{1-q}\right)$.*

(c) *For every key $x \in S$, the probability that $(h_1, h_2) \in R(S)$ and that in the graph obtained from $G(S, h_1, h_2)$ by removing the edge $(h_1(x), h_2(x))$ the vertex $h_1(x)$ is the endpoint of a simple path of length t is bounded by $(1+\epsilon)^{-t}$.*

3 Proof of the Main Theorem

3.1 Some graph theoretical observations

Recall the definition of the cyclomatic number of a graph G. An edge e is called a *leaf edge* if it has an endpoint with degree 1; the non-leaf edges are called *inner edges*. An edge is called a *cycle edge* if it lies on a cycle.

Our goal is to determine upper bounds on the probability that a randomly determined bipartite graph is isomorphic to a fixed connected graph with a given cyclomatic number (and a given number of leaf edges). In order to achieve this, we need an upper bound on the number of such graphs. Let $N(k, \ell, q)$ be the number of non-isomorphic connected graphs whose cyclomatic number is q and which have $k - \ell$ inner edges and ℓ leaf edges. (By Lemma 1, such graphs have $k - q + 1$ nodes.) Further, let $N^*(k, p)$ be the number of non-isomorphic connected graphs with k edges, p of which are either cycle or leaf edges.

Lemma 2.

(a) $N(k, \ell, 0) \;\leq\; k^{2\ell-4}$.

(b) $N(k, \ell, q) \;=\; k^{O(\ell+q)}$.

(c) $N^*(k, p) \;=\; k^{O(p)}$.

Proof. (a) In case $q = 0$ the graphs are trees. Any tree with $k - \ell$ inner edges and $\ell \geq 2$ leaf edges can be constructed in the following way. One starts with a path of length $k_2 \in [2, k - (\ell - 2)]$, called G_2 (the index refers to the number of leaf edges in the graph). Then for $i = 3, \ldots, \ell$, one constructs G_i from G_{i-1} by taking a new path of length $k_i \geq 1$ such that $k_2 + \cdots + k_i \leq k - (\ell - i)$ and identifying one endpoint of this path with an arbitrary vertex in G_{i-1}. The length k_ℓ of the last path is uniquely determined by $k_\ell = k - (k_2 + \cdots + k_{\ell-1})$. There are fewer than $k^{\ell-2}$ choices for picking the lengths $k_2, \ldots, k_{\ell-1}$. Furthermore, in each of the $\ell - 2$ steps, where G_i is constructed from G_{i-1}, there are fewer than k possibilities for the vertex at which the new path is attached. Hence, $N(k, \ell, 0)$ is bounded by $k^{2\ell-4}$.

(b) The case $q = 0$ was treated in (a). Now let $q > 0$ and consider the connected graphs G with $k - \ell$ inner edges and ℓ leaf edges and cyclomatic number q. Note that Definition 3 implies that starting with G we can remove q cycle edges in such a way that a spanning tree is left. This tree has $\ell' \leq \ell + 2q$ leaf edges. Since each of the q cycle edges has no more than k^2 possibilities for its endpoints, we may use (a) to estimate

$$\begin{aligned} N(k, \ell, q) &\leq k^{2q} \cdot N(k - q, \ell', 0) \\ &\leq k^{2q} \cdot (k - q)^{2\ell+4q-4}. \end{aligned}$$

(c) Let $N^*(k, p, q)$ be the number of non-isomorphic connected graphs whose cyclomatic number is q and which have k edges, p of which are either cycle edges or leaf edges. Clearly, $N^*(k, p, 0) = N(k, p, 0) \leq k^{2p-4}$, by (a). For $q > 0$ consider all connected graphs G with k edges, p of which are either cycle edges or leaf edges, and cyclomatic number q. Just as in the proof of part (a), from such a graph G we may remove q cycle edges one after another to obtain a spanning tree T with $k - q$ edges. Now note that if an edge e' becomes a leaf edge when the cycle edge e is removed, then e' must lie on the fundamental cycle created by e and T. This means that all leaf edges in T were cycle edges or leaf edges in G, hence T has no more than $p - q$ leaf edges. (Note that it is in fact possible that $p - q$ leaf edges remain, if the original graph contains no leaf edges and all cycles are 2-cycles.) As before, the number of ways to place the q cycle edges is bounded by k^{2q}, hence, by (a),

$$\begin{aligned} N^*(k, p, q) &\leq k^{2q} \cdot N^*(k - q, p - q, 0) \\ &= k^{2q} \cdot N(k - q, p - q, 0) \\ &\leq k^{2q}(k - q)^{2(p-q)-4} \; < \; k^{2p-4}. \end{aligned}$$

Summing up the upper bound for $N^*(k, p, q)$ for all $0 \leq q \leq k$ yields the claimed upper bound on $N^*(k, p)$. $\square$

We can now state an upper bound on the probability that a (truly) random bipartite graph is isomorphic to a connected graph with a given cyclomatic number. Let as in Definition 2 $G = G(S, h_1, h_2)$ be the bipartite graph determined by two hash functions $U \to [m]$ and a set $S \subseteq U$. For $T \subseteq S$ let $K(T) = G(T, h_1, h_2)$ be the subgraph of G consisting of the edges that belong to keys in T, disregarding the isolated nodes. We assume that each edge $(h_1(x), h_2(x))$, $x \in T$, in the graph $K(T)$ is labeled with the key x. We say that two graphs $H = (V_H, E_H)$ and $H' = (V_{H'}, E_{H'})$ with labeled edges are isomorphic if there are bijections $\sigma : V_H \to V_{H'}$ and $\tau : E_H \to E_{H'}$ (as multisets) such that if $e \in E_H$ connects $v, w \in V_H$, then $\tau(e)$ connects $\sigma(v), \sigma(w)$, and such that for all $e \in E_H$ the labels of e and $\tau(e)$ are the same.

Lemma 3. *Let $T \subseteq U$, and $H = (V_H, E_H)$ be a bipartite, connected graph, where each edge is labeled with a unique element in T. If the values $h_i(x)$ are chosen fully randomly for all $x \in T, i \in \{1, 2\}$, then the probability that $K(T)$ is isomorphic to H is at most $2 \cdot m^{-|E_H|-q+1}$, where q is the cyclomatic number of H.*

Proof. Consider a vertex v of the graph H that is adjacent to d edges marked with $x_1, \dots, x_d \in S$. If H is isomorphic to $K(T)$, then there exists $i \in \{1,2\}$ such that $h_i(x_1) = \cdots = h_i(x_d)$. Since H is bipartite and connected, there is exactly one way to split V_H into color classes U_H and W_H such that $E \subseteq U_H \times W_H$. Thus, there are two ways to color the vertices with colors 1 and 2. The probability that $K(T)$ is isomorphic to H is bounded by the probability that there exists a coloring of the vertices of H such that for all vertices v we have the following: if v is colored i and v is incident to edges labeled with $x_1, \dots, x_{d_v} \in S$ then we have $h_i(x_1) = \cdots = h_i(x_{d_v})$. The probability that this is true for one node v is exactly m^{-d_v+1}, due to the independence of all random values $h_i(x)$. Hence, taking the two possible ways of coloring the vertices into account, the probability that H is isomorphic to $K(T)$ is at most $2 \cdot \prod_{v \in V_H} m^{-\deg(v)+1} = 2 \cdot m^{|V_H|-D}$, where $D = \sum_{v \in V_H} \deg(v) = 2|E_H|$. The claim now follows from $|V_H| = |E_H| - q + 1$ as stated in Lemma 1. $\square$

3.2 Bad subgraphs

Now we turn to the situation that underlies Theorem 1, i. e., we consider the graphs generated by choosing a pair (h_1, h_2) of hash functions from $\hat{\mathcal{R}}^d_{r,m}$, see Definition 1. (Recall that each hash function $h_i = h_{f_i,g,z^{(i)}}$ is determined by d-wise independent hash functions f_i and g and a vector $z^{(i)}$ of random offsets by $h_i(x) = \left(f_i(x) + z^{(i)}_{g(x)}\right) \bmod m$.) The statements made in Theorem 1 refer to a partition of $\hat{\mathcal{R}}^d_{r,m}$ into a set $R(S)$ containing pairs which are "good" for S and the remaining pairs which are "bad". We identify now the set of good hash function pairs.

Definition 4. *Let $S \subseteq U$.*

(a) *For $T \subseteq S$, the set $R^*(T)$ consists of those hash function pairs (h_1, h_2) whose g-component satisfies $|g(T)| \geq |T| - \ell$.* [2]

(b) *$G = G(S, h_1, h_2)$ is* ℓ-bad *if there exists a subset $T \subseteq S$ such that $K(T)$ is connected and $(h_1, h_2) \notin R^*(T)$.*

(c) *$R(S) \subseteq \hat{\mathcal{R}}^{2\ell}_{r,m}$ is the set of those hash function pairs (h_1, h_2) for which the graph $G(S, h_1, h_2)$ is not ℓ-bad.*

We can now formulate a quite general statement that captures a central property of $R^*(T)$. It makes precise the idea that in the random graphs $G(S, h_1, h_2)$ everything inside the connected components is practically random. This result is also a building block of the proof of Theorem 1, which will be given in the next section.

[2]If one regards g as a hash function from U to r that maps keys to "buckets" $0, 1, \dots, r-1$, then this means that the keys in T cover at least $|T| - \ell$ buckets of g, that means, g distributes the keys of T rather well.

Theorem 2. *As in Theorem 1, let $\epsilon > 0$ and $\ell \geq 1$ be fixed, let $m = (1+\epsilon)n$, and consider some $S \subseteq U$, $|S| = n$. Define $R(S) \subseteq \hat{\mathcal{R}}^{2\ell}_{r,m}$ as above. Then for each subset $T \subseteq S$ the set $R^*(T)$ of hash function pairs has the following properties:*

(a) *If $(h_1, h_2) \in R(S)$ and the edge set $K(T) = \{(h_1(x), h_2(x)) \mid x \in T\}$ is contained in a connected component of $G = G(S, h_1, h_2)$, then $(h_1, h_2) \in R^*(T)$.*

(b) *If (h_1, h_2) is chosen at random from $R^*(T)$ then $(h_1(x), h_2(x))$, $x \in T$, is uniformly and independently distributed in $[m]^2$.*

Put a little more informally, this means that if we are interested in the behaviour of the hash functions (h_1, h_2) on arbitrary key sets T, but only as long as all edges labeled with keys from T lie in the same connected component, then for each T only the subset $R^*(T)$ of the hash function pairs is of interest. Within this set the random experiment for choosing (h_1, h_2) distributes the edges fully randomly, with the possible exception of the intersection $R^*(T) \cap \left(\hat{\mathcal{R}}^{2\ell}_{r,m} - R(S)\right) \subseteq \hat{\mathcal{R}}^{2\ell}_{r,m} - R(S)$, a set not depending on T, which in Theorem 1 is shown to be very small.

3.3 The proofs

Proof of Theorem 2. (a) Suppose for a contradiction that $(h_1, h_2) \notin R^*(T)$, i. e., $|g(T)| < |T| - \ell$. By assumption, $K(T)$ is contained in a connected component K' of G, which can be written $K' = K(T')$ for some T' with $T \subseteq T'$. Obviously, then $|g(T')| < |T'| - \ell$, which means that $G(S, h_1, h_2)$ is ℓ-bad, contradicting the assumption that (h_1, h_2) is in $R(S)$.

(b) Assume that (h_1, h_2) is chosen randomly from $R^*(T)$, i. e., from the set of function pairs for which $|g(T)| = |T| - \ell'$ for some $\ell' \leq \ell$. First we fix such a g. Then we can pick $|T| - \ell'$ keys from T such that they are all hashed by g into different buckets while the remaining ℓ' keys are hashed into at most ℓ' of these buckets. Hence, we can partition T into two sets T_1, T_2 such that $|T_1| = \ell'$, $|g(T_2)| = |T_2| = |T| - \ell'$, and $g(T_1) \subseteq g(T_2)$. Let $T_2' \subseteq T_2$ be the set of at most ℓ' keys for which $g(T_1) = g(T_2')$, and note that $|T_1 \cup T_2| \leq 2\ell$. We now choose f_1 and f_2 and the random offsets in a particular order. First choose the offsets $z_i^{(1)}$ and $z_i^{(2)}$ randomly for all $i \in g(T_1)$, and assume they are fixed. Then still, the hash values $h_1(x) = \left(f_1(x) + z^{(1)}_{g(x)}\right) \bmod m$ and $h_2(x) = \left(f_2(x) + z^{(2)}_{g(x)}\right) \bmod m$ for $x \in T_1 \cup T_2'$ are distributed independently and uniformly in $[m]$ because f_1 and f_2 are 2ℓ-wise independent. Now note that $g(T_2 - T_2') \cap g(T_1 \cup T_2') = \emptyset$, and that $|g(T_2)| = |T_2|$. Hence the offsets $z_i^{(j)}$ with $j \in \{1, 2\}$ and $i \in g(T_2 - T_2')$ can now be chosen independently from each other and from those with $i \in g(T_1 \cup T_2')$. This implies that also the hash values $h_i(x)$ with $i \in \{1, 2\}$ and $x \in T_2 - T_2'$ are distributed independently and uniformly, and also independently from the values for $x \in T_1 \cup T_2'$. □

Lemma 4. *Let $G = G(S, h_1, h_2)$. If G is ℓ-bad, then there exists a subset $T \subseteq S$ such that $|g(T)| = |T| - \ell$, $K(T)$ is connected, and the total number of leaf and cycle edges in $K(T)$ is at most 2ℓ.*

Proof. Let $T \subseteq S$ and $K = K(T)$. An edge $(h_1(x), h_2(x))$ in K is called *special* if there is some other key $x' \in T$, $x \neq x'$, such that $g(x) = g(x')$.

Since G is ℓ-bad, there is some set $T \subseteq S$ such that $K = K(T)$ is connected and $|g(T)| < |T| - \ell$. We iteratively delete keys from T, and the corresponding edges from K, as long as this can be done in such a way that the remaining subgraph $K' = K(T')$ is connected and for the remaining set T' of keys we have $|g(T')| \le |T'| - \ell$.

When this process stops with a key set T' remaining then all leaf edges and all edges in cycles must be special because non-special leaf or cycle edges would have been removed by the process. Further, $|g(T')| = |T'| - \ell$ because otherwise one could remove one special leaf or cycle edge. This implies that there is a subset $T'' \subseteq T'$ of size ℓ such that each key in T'' collides under g with a key not in T'', while all the keys in $T' - T''$ are hashed by g into different buckets. Hence, exactly the edges corresponding to keys in T'' and at most $|T''|$ edges corresponding to keys in T' are special. Thus, $K' = K(T')$ has at most 2ℓ special edges. To conclude, $K(T')$ contains at most 2ℓ leaf and cycle edges and $|g(T')| = |T'| - \ell$. □

Proof of Theorem 1. (a) We mark each edge $(h_1(x), h_2(x))$, $x \in S$, of $G = G(S, h_1, h_2)$ with the corresponding key x. If G is ℓ-bad, then according to Lemma 4 there exists a subset $T \subseteq S$ such that $|g(T)| = |T| - \ell$ and $K(T)$ contains at most 2ℓ leaf and cycle edges.

Let H be a connected bipartite graph with k edges which are marked uniquely with the elements from a k-element set $T \subseteq S$. Further, let H have at most 2ℓ leaf and cycle edges. We bound the probability of the event $I(H)$, which is that $K(T)$ is isomorphic to H and that $|g(T)| = |T| - \ell$.

If $|g(T)| = |T| - \ell$, then T can be partitioned into two disjoint subsets T_1, T_2 such that $|g(T_2)| = |T_2| = |T| - \ell$ and $g(T_1) \subseteq g(T_2)$. In this case there is a subset $T_2' \subseteq T_2$ with $1 \le |T_2'| \le \ell$ such that $g(T_1) = g(T_2')$ and $|g(T_2')| = |T_2'|$. There are fewer than $k^{2\ell}$ possibilities to choose the subsets T_1 and T_2' of T, and due to the 2ℓ-wise independence the probability of $g(T_1) = g(T_2')$ and $|g(T_2')| = |T_2'|$ is bounded above by $(|T_2'|/r)^{|T_1|} \le (\ell/r)^\ell$. Hence, the probability that $|g(T)| = |T| - \ell$ is bounded by $(k^2\ell/r)^\ell$.

If this event occurs, then the hash values $h_i(x)$ with $x \in T$ and $i \in \{1, 2\}$ are independently and uniformly distributed which can be verified along the lines of the proof of Theorem 2 (b). Hence, in this case, according to Lemma 3 the probability that $K(T)$ is isomorphic to H is at most $2m^{-k+1}$. Overall, we obtain

$$\mathbf{Prob}\big(I(H)\big) \;\le\; (k^2\ell/r)^\ell \cdot 2m^{-k+1} .$$

According to Lemma 2, there are $N^*(k,p) = k^{O(p)}$ possibilities to choose a bipartite graph H with k edges among which there are p leaf and cycle edges. Since ℓ is a constant, there are $k^{O(1)}$ possibilities to choose H in such a way that it contains at most 2ℓ leaf and cycle edges. Furthermore, there are fewer than n^k ways to choose the set T of k keys and to mark the edges of H uniquely with the elements from T. Summing over all k, we obtain the following upper bound on the probability that $G(S, h_1, h_2)$ is ℓ-bad:

$$\begin{aligned}\mathbf{Prob}\big(\exists H : I(H)\big) \;&=\; \sum_{k=2}^{n} k^{O(1)} n^k (k^2\ell/r)^\ell \cdot 2m^{-k+1} \\ &=\; r^{-\ell} \cdot \sum_{k=2}^{n} k^{O(1)} \frac{n^k}{m^{k-1}} \;=\; \frac{n}{r^\ell} \cdot \sum_{k=2}^{n} \frac{k^{O(1)}}{(1+\epsilon)^{k-1}} \;=\; O\left(\frac{n}{r^\ell}\right).\end{aligned}$$

(b) If $G = G(S, h_1, h_2)$ contains a connected component with cyclomatic number at least q, then there is a connected subgraph H of G whose cyclomatic number is q and which does not contain any leaf edges. This subgraph can be obtained by repeated deletion of cycle edges until the cyclomatic number is q (Lemma 1 implies that the deletion of a cycle edge reduces the cyclomatic number by 1) and afterwards by repeated deletion of leaf edges until no leaf edges remain. Note that deleting a leaf or a cycle edge does not destroy connectivity.

We prove the claimed probability bound for the event that $(h_1, h_2) \in R(S)$ and that there exists a subset $T \subseteq S$ of the keys such that $K(T)$ is a connected subgraph of G with cyclomatic number q and which does not contain any leaf edges. According to Theorem 2 (a), the subgraph $K(T)$ can only be connected for $(h_1, h_2) \in R(S)$ if $(h_1, h_2) \in R^*(T)$. Hence, by Theorem 2 (b) it suffices to bound the probability that there is $T \subseteq S$ such that $K(T)$ is connected, contains no leaf edges, and has cyclomatic number q under the assumption that all edges of $K(T)$ are chosen completely at random.

There are $N(k, 0, q) = k^{O(1)}$ (see Lemma 2(b)) possibilities to choose a connected bipartite graph with k edges, no leaves, and cyclomatic number q. There are fewer than n^k possibilities to choose $T \subseteq S$, $|T| = k$, and to label the edges of H with the elements in T. Hence, according to Lemma 3 the probability that there exists $T \subseteq S$, $|T| = k$, such that $K(T)$ is isomorphic to a graph with cyclomatic number q and no leaf edges is bounded by $k^{O(1)} \cdot n^k \cdot 2m^{-k-q+1}$. Summing over all possible values of k we obtain an upper bound of

$$\sum_{k=q+1}^{n} n^k k^{O(1)} m^{-k-q+1} = n^{-q+1} \sum_{k=q+1}^{n} \frac{k^{O(1)}}{(1+\epsilon)^{k+q-1}}$$

for the probability that $(h_1, h_2) \in R(S)$ and that G has a connected component with cyclomatic number at least q. Since this is $O(n^{-q+1})$, the claim follows.

(c) If $h_1(x)$ is the endpoint of a simple path of length t that does not contain the edge $e = (h_1(x), h_2(x))$, then there exist t elements $x_1, \ldots, x_t \in S - \{x\}$ such that the path consists of the edges $e_1, \ldots, e_t$, where $e_i = (h_1(x_i), h_2(x_i))$. Let $x_0 = x$. Since $h_1(x_0)$ and $h_1(x_1)$ are both endpoints of e_1, and e_i is adjacent to e_{i+1}, we have $h_1(x_0) = h_1(x_1) \wedge h_2(x_1) = h_2(x_2) \wedge \ldots$. More precisely,

$$\forall 0 \le i \le t-1: \;\; h_{i \bmod 2+1}(x_i) = h_{i \bmod 2+1}(x_{i+1}). \tag{1}$$

Hence we have to bound the probability that (1) is true and $(h_1, h_2) \in R(S)$. Let $T = \{x_0, x_1, \ldots, x_t\}$ and note that the event (1) may occur only if $K(T)$ is connected. Hence, similar as in the proof of part (b), it suffices according to Theorem 2 to bound the probability of (1) under the assumption that the random values $h_i(x)$, $x \in T = \{x_0, x_1, \ldots, x_t\}$, $i \in \{1, 2\}$, are all independently and uniformly distributed. In this case the probability of (1) is exactly m^{-t}. Since there are fewer than n^t possible choices for $x_1, \ldots, x_t$, the probability that such a path exists and $(h_1, h_2) \in R(S)$ is at most $(n/m)^t = (1+\epsilon)^{-t}$. □

4 Applications

4.1 Cuckoo hashing without Siegel's functions

In this section we show that a variant of the analysis of cuckoo hashing given in [19] remains valid if function pairs from $\hat{\mathcal{R}}^4_{r,m}$ are used. We consider the version of the cuckoo hashing data structure as described in Section 1.2, and analyze its operation during a phase that comprises ρn update operations. Here, $\rho > 0$ is constant, and $n \geq |S|$ for the set $S \subseteq U$ that consists of all keys that are stored in the table at any time during the phase. All hash function pairs (h_1, h_2) are chosen at random from $\hat{\mathcal{R}}^4_{r,m}$, where $m = (1+\epsilon)n$ and $r = m^\delta$, $\frac{1}{2} < \delta < 1$ constant. The bound L is chosen to be $\Theta(\log n)$. The phase starts with some set of keys being stored in the dictionary according to some hash function pair (h_1, h_2). In the analysis, we disregard lookup and deletion operations. Our aim is to show that if we perform the insertion operations by the procedure described in Section 1.2, then the expected time for these insertions is $O(n)$.

We split the phase into subphases. The first subphase starts with the beginning of the phase, subsequent subphases start whenever a new pair (h_1, h_2) of hash functions is chosen (and the data structure is built anew). Thus a subphase is exactly the lifespan of a hash function pair.

Lemma 5. *The total time spent for insertions during a subphase is $O(n \log n)$ in the worst case.*

Proof. The insertion procedure aborts and starts a new subphase after L keys have become nestless in one attempt to insert a key. Thus no insertion attempt takes longer than $O(\log n)$ time; and even if the dictionary is built anew, no more than $(1+\rho)n$ keys are inserted in a subphase. □

Lemma 6. *The expected number of subphases is $1 + O(n/r^2)$.*

Proof. It is sufficient to show that for each subphase the probability that it ends before the end of the phase is bounded by $O(n/r^2)$. Let (h_1, h_2) be the pair of hash functions used in the subphase. The probability that $(h_1, h_2) \notin R(S)$ is $O(n/r^2)$, by Theorem 1(a). Thus, we can assume that $(h_1, h_2) \in R(S)$. We have to estimate the probability that the subphase ends prematurely. For this to happen, there must be a key x inserted during the subphase that causes a sequence of L "nestless" keys. For this, there are two possibilities:
Case 1: The set of edges in $G(S, h_1, h_2)$ determined by these nestless keys forms a subgraph with cyclomatic number $q \geq 2$. Then it is immediate that $G(S, h_1, h_2)$ has a connected component with cyclomatic number q. Theorem 1(b) implies that the probability that this happens (while $(h_1, h_2) \in R(S)$) is $O(n^{-1}) = O(n/r^2)$.
Case 2: The set of edges in $G(S, h_1, h_2)$ determined by these L nestless keys determines a subgraph with cyclomatic number at most 1. It is then easy to see, similarly as in the corresponding proof in [19, Section 3.1], that this subgraph contains a simple path starting at node $h_1(x)$ or at $h_2(x)$ of length $(L-1)/3$, which does not contain the edge $(h_1(x), h_2(x))$ corresponding to x. Theorem 1(c) says that the probability that this happens is bounded by $2(1+\epsilon)^{-(L-1)/3}$. If we let $L \geq 6\lceil \log_{1+\epsilon}(n) \rceil + 1$, this is at most $2n^{-2}$. Thus the probability that this occurs for *some* key x is at most $2(1+\rho)n^{-1} = O(n/r^2)$. □

Lemmas 5 and 6 imply that the expected time spent in subphases that end prematurely (i. e., all excepting the last subphase) is $O((n/r^2) \cdot n \log n) = o(n)$. Further, arguing in exactly the same way we see that the expected time we spend in a last subphase that uses a pair $(h_1, h_2) \notin R(S)$ is $O((n/r^2) \cdot n \log n) = o(n)$. This means that from here on we may concentrate on the last subphase, in which all insertions are successfully completed, and may assume that a hash function pair $(h_1, h_2) \in R(S)$ is used. We can argue just as in [19, Section 3.1]: For each key x inserted during this subphase we have the following: If it creates a sequence of t or more nestless keys, then $G(S, h_1, h_2)$ contains a simple path of length at least $(t-1)/3$, starting with $h_1(x)$ or at $h_2(x)$, and not containing the edge $(h_1(x), h_2(x))$. The probability for this to happen is (assuming $(h_1, h_2) \in R(S)$) bounded by $2(1+\epsilon)^{-(t-1)/3}$, by Theorem 1(c). This implies that the expected number of nestless keys generated during the insertion of x is at most $\sum_{t \geq 1} 2(1+\epsilon)^{-(t-1)/3} = O(1)$, hence the expected insertion time is $O(1)$ in this case. — Thus we have proved the following theorem.

Theorem 3. *Assume that (h_1, h_2) is chosen at random from $\hat{\mathcal{R}}^4_{r,m}$, where $m = (1+\epsilon)n$ and $r = m^\delta$, for $\frac{1}{2} < \delta < 1$ constant. Assume further that a phase involving at most n keys and ρn updates is run as described. Then lookups and deletions can be carried out in worst case constant time, and insertions in amortized expected constant time.*

4.2 Uniform hashing

In this section we show how we can exploit the property of our random graphs $G(S, h_1, h_2)$ to have a constant bound on the cyclomatic number of each connected component in order to simulate uniform hashing on S.

Let in the following $t \geq 1$ and let $\oplus$ be an arbitrary group operation over $[t]$ (e.g. addition modulo t).

Definition 5. *Let $m \in \mathbb{N}$ and $h_1, h_2 : U \to [m]$. For a vector $\phi = (\phi_1, \ldots, \phi_m)$ of m hash functions $U \to [t]$ and a vector $\lambda = (\lambda_1, \ldots, \lambda_m)$ of m offsets in $[t]$ the hash function $\alpha := \alpha_{h_1,h_2,\phi,\lambda} : U \to [t]$ is defined by*

$$\alpha(x) \;=\; \phi_{h_1(x)}(x) \oplus \lambda_{h_2(x)}.$$

For each pair (h_1, h_2) and $c \geq 2$ the hash class $\mathcal{F}_{c,h_1,h_2}$ consists of all hash functions $\alpha = \alpha_{h_1,h_2,\phi,\lambda}$, where $\phi_1, \ldots, \phi_m \in \mathcal{H}^c_t$ and $\lambda_1, \ldots, \lambda_m \in [t]$.

The behaviour of a function chosen at random from the class $\mathcal{F}_{c,h_1,h_2}$ on $S \subseteq U$ is best described in our graph theoretical setting from above. Consider the bipartite graph $G = G(S, h_1, h_2) = (V \uplus W, E)$, where V and W are disjoint copies of $[m]$. With each vertex $v \in V$ we associate a hash function ϕ_v randomly chosen from the c-wise independent class $\mathcal{H}^c_t$, and with each vertex $w \in W$ we associate a randomly chosen offset $\lambda_w \in [t]$. Each key $x \in S$ is associated with the edge $e(x) = \big(h_1(x), h_2(x)\big)$. Hence we may identify the edge set E with the key set S. Then the behaviour of the hash function α on S is equivalent to the behaviour of the hash function $\beta : E \to [t]$, $(v, w) \mapsto \phi_v((v, w)) \oplus \lambda_w$, on E. Therefore, the hash class $\mathcal{F}_{c,h_1,h_2}$ given by (h_1, h_2) is uniform on S if and only if the hash values $\beta(e)$, $e \in E$, are uniformly and independently distributed. The following theorem shows that this is the case if the cyclomatic number of every connected component in G is bounded by $c/2$.

Theorem 4. *Let $G = (V \uplus W, E)$ be a bipartite graph such that the cyclomatic number of every connected component of G is bounded by $c/2$, $c \in \mathbb{N}$. Let $\beta : E \to [t]$ be determined as described above by independently and randomly choosing $\phi_v \in \mathcal{H}_t^c$ and $\lambda_w \in [t]$ for $v \in V$ and $w \in W$. Then the hash values $\beta(e)$, $e \in E$, are uniformly and independently distributed.*

Before we give the proof of this theorem, we note some consequences. By the discussion above and according to Theorem 1 Theorem 4 implies that if we choose $(h_1, h_2) \in \hat{\mathcal{R}}_{r,m}^d$ at random then the hash class $\mathcal{F}_{c,h_1,h_2}$ is uniform for S with high probability:

Corollary 1. *Let $\epsilon > 0$ and $\ell, c \in \mathbb{N}$ be fixed and let $m = m(n) = (1 + \epsilon)n$. For any set $S \subseteq U$, $|S| = n$, and randomly chosen $(h_1, h_2) \in \hat{\mathcal{R}}_{r,m}^{2\ell}$ the probability that $\mathcal{F}_{c,h_1,h_2}$ is uniform on S is at least $1 - n/r^\ell - O\big(n^{-\lfloor c/2 \rfloor}\big)$.*

For example, for given n we could choose $r = n/\log n$ and $c = \ell = 2$, and use linear hash functions as ϕ_w (see [5]); for the 2ℓ-wise independent hash classes needed for $\hat{\mathcal{R}}_{r,m}^{2\ell}$ we can take the classes to be presented in Section 5. Then the total space required for describing $(h_1, h_2) \in \hat{\mathcal{R}}_{r,m}^{2\ell}$ can be made $o(n)$; the space required for the function from $\mathcal{F}_{c,h_1,h_2}$ can be bounded by $2(1 + \epsilon)n$ words from U (for describing $\phi_1, \dots, \phi_m$) plus $(1 + \epsilon)n$ numbers from $[t]$ (bit length $\log t$). Further, each hash function from this class can be evaluated by two multiplications of $O(\log |U|)$-bit integers, two additions, and some table-lookups and bit operations. The probability that $\mathcal{F}_{c,h_1,h_2}$ is not uniform for a set S, $|S| = n$, is bounded by $1/n^{1-\delta}$ for arbitrarily small $\delta > 0$. By increasing the parameters ℓ and c, one may decrease the failure probability to an arbitrarily small polynomial in n^{-1}, and still have constant evaluation time and linear space. Note, though, that increasing c to obtain a better reliability also increases the space requirements by a factor of $O(c)$.

In order to prove Theorem 4 we need a graph theoretical lemma relating the cyclomatic number of a graph G to the degree of vertices in the subgraph of G which consists only of cycles and of paths connecting cycles. As before, we call an edge a *cycle edge*, if it lies on a cycle. An edge that is not a cycle edge but lies on some simple path whose endpoints are both adjacent to cycle edges is called *connecting edge*. Cycle edges and connecting edges together form the "cyclic part" or "2-core" of G, which is studied from a different viewpoint in [15].

Lemma 7. *If the cyclomatic number of every connected component of a graph $G = (V, E)$ is bounded by q, then each vertex $v \in V$ is adjacent to at most $2q$ edges that are cycle or connecting edges.*

Proof. Fix a vertex v and a spanning tree T of the connected component of v. The edges $e_1, \dots, e_\rho \in T$ that are incident with v connect v to disjoint trees $T_1, \dots, T_\rho$. We now add the $\leq q$ cycle edges of the component missing in T one by one. When we add an edge that connects two vertices inside one subtree T_σ, then e_σ may turn into a connecting edge, all other edges incident to v keep their status. If we add an edge that connects v with a node in T_σ, then e_σ turns into a cycle edge, the other edges of v are unaffected. Finally, if we add an edge that connects two nodes in different subtrees T_σ and T_τ, then e_σ and e_τ become cycle edges, the other edges of v are unaffected. Thus, if v is incident with ℓ connecting edges and k cycle edges, then $\ell + k/2 \leq q$; in any case we have $k + \ell \leq 2q$. $\square$

of Theorem 4. Note that an offset λ_w can be considered as the constant hash function $E \to [t], e \mapsto \lambda_w$. Hence, for a vertex $u \in V \uplus W$ we call ϕ_u (if $u \in V$) or λ_u (if $u \in W$) the *hash function associated with u*.

We consider two edge disjoint subgraphs $G_C = (U_C, E_C)$ and $G_F = (U_F, E_F)$ of G. The edge set E_C consists of all cycle and connecting edges of G and E_F consists of all other edges. $U_C = V_C \uplus W_C$ and $U_F = V_F \uplus W_F$ are the sets of all vertices which are adjacent to at least one of the edges in E_C and E_F, respectively. Note that while E_C and E_F form a partition of the edge set, U_C and U_F are not necessarily disjoint. — We prove the following two statements.

(i) Choosing the hash functions associated with the vertices in U_C independently at random yields an independent and uniform distribution of the hash values $\beta(e)$ with $e \in E_C$.

(ii) Assuming that the hash functions associated with the vertices in U_C are fixed, then choosing the hash functions associated with the vertices in $U_F - U_C$ independently at random yields an independent and uniform distribution of the hash values $\beta(e)$ with $e \in E_F$.

Note that the random choice of the hash functions associated with the vertices in $U_F - U_C$ has no influence on the hash values $\beta(e)$ with $e \in E_C$, because no such edge is adjacent to a vertex in $U_F - U_C$. Clearly, the theorem follows from (i) and (ii).

Proof of (i): First, randomly choose λ_w, for $w \in W_C$, and consider these values as fixed from here on. Now consider some fixed $v \in V_C$. Let $q = \lfloor c/2 \rfloor$; by the assumption in the theorem the cyclomatic number of the connected component of v is bounded by q. By Lemma 7, it follows that v is incident to at most $2q \leq c$ edges in E_C. Hence, the hash values $\beta(e) = \phi_v(e) \oplus \lambda_w$ of all edges $e = (v, w) \in E_C$ which are adjacent to v are uniformly and independently distributed, because ϕ_v is chosen from a c-wise independent hash family. Since each edge in E_C is incident to exactly one vertex $v \in V_C$, and since all hash functions ϕ_v are chosen independently, we conclude that all hash values $\beta(e)$, $e \in E_C$, are uniformly and independently distributed.

Proof of (ii): By definition, G_F is a forest. Moreover, each tree in G_F contains at most one vertex in U_C: If there were two different vertices $u, u' \in U_C$ in the same tree of G_F, then the path in G_F connecting u and u' would either belong to a cycle or it would connect two cycles. But then this path would either consist entirely of cycle edges or connecting edges in contradiction to the definition of E_F.

The proof is by induction on the number of edges in the forest G_F. If G_F contains no edge, then there is nothing to show. Hence, assume that G_F contains $k \geq 1$ edges. Since each tree has at least two leaves and at most one vertex of each tree is in U_C, we can choose a leaf u such that $u \in U_F - U_C$. Let $e^* = (u', u)$ be the unique edge adjacent to u and assume w.l.o.g. that $u \in W$ (the other case is analogous). Then $\beta(e^*) = \phi_{u'}(e^*) + \lambda_u$. If we remove e^* and u from G_F, then we are left with a forest containing $k-1$ edges, and by the induction hypothesis, choosing the hash functions associated with the vertices in $U_F - \{u\}$ uniformly at random yields an independent distribution of the hash values $\beta(e)$, $e \in E_C - \{e^*\}$. On the other hand, the random choice of λ_u yields a uniform distribution of $\beta(e^*)$ which is independent of the hash values $\beta(e)$, $e \in E_C - \{e^*\}$, because u is only adjacent to e^*. Hence, all hash values $\beta(e)$, $e \in E_C$, are uniformly and independently distributed. $\square$

4.3 Simulating shared memory

For illustrating the use of our function pairs in the context of simulations of shared memory on distributed memory machines we just give one example. In [11, p. 530] a "Process_3" is described. It assumes a key set S is given and functions h_1, $h_2 : U \to [m]$ are chosen. The process is an abstraction of a method for retrieving a copy of x from memory module $h_1(x)$ or from memory module $h_2(x)$ for each $x \in S$, using a memory module only for one key in one round. One proceeds in rounds $t = 1, 2, \ldots$. In round t, for each $j \in [m]$ in parallel, first an $x \in h_1^{-1}(\{j\})$ is removed from S, then an $x \in h_2^{-1}(\{j\})$ (if such keys exist). Theorem 6.4 in [11] says that this process will have emptied set S after $O(\log \log n)$ rounds with high probability. The proof of this statement solely relies on observations on the size and the cycle structure of the connected components of $G(S, h_1, h_2)$. The same statements follow from Theorems 1 and 2, if our class $\hat{\mathcal{R}}^d_{r,m}$ is used.

5 d-wise independence without polynomials

In this section, we describe how approximate d-wise independence can be achieved without the use of polynomials or of expander graphs. It even turns out that (exploiting word-level parallelism) one integer multiplication and some bit operations and table lookups are sufficient to realize an approximately d-wise independent class. The drawback is that storing a function from the class takes much more space than in the classical constructions (but still comparable to the space used by Siegel's functions). These approximately d-independent classes may be used as building block for the function pair classes considered in previous sections.

5.1 The basic construction

Let $(M, \oplus)$ be a finite abelian group, and let $m = |M|$. M will serve as the range of our hash functions. Simple examples for M would be $\{0,1\}^\mu$ for $m = 2^\mu$, with $\oplus$ being bitwise XOR, or $\{0, 1, \ldots, m-1\}$, for any m, with $\oplus$ being addition modulo m.

Let $0 < \delta < 1$ be an arbitrary constant, and let $r = m^\delta$. Assume $c \geq 1$ is constant and $\mathcal{H}$ is an arbitrary c-universal class of hash functions from the universe U to $R = \{1, \ldots, r\}$. (I.e., for x, y different elements of U and f chosen randomly from $\mathcal{H}$ we have $\Pr(f(x) = f(y)) \leq c/r$. See, e. g., [2] for the existence of such classes.) Let $d \geq 2$ be an arbitrary constant. For arbitrary fixed $\ell \geq 1$, let $m_{i,s}$, $1 \leq i \leq \ell$, $1 \leq s \leq r$, be randomly chosen elements of M. Further, let $f_1, \ldots, f_\ell$ be chosen at random from $\mathcal{H}$. Define

$$h(x) = m_{1,f_1(x)} \oplus \cdots \oplus m_{\ell,f_\ell(x)}.$$

Then, for $\ell = O(d)$ sufficiently large, h behaves like a function from an approximately d-independent class, in the following sense.

Theorem 5. *Let $x_1, \ldots, x_d$ be arbitrary distinct elements of U, and let $a_1, \ldots, a_d$ be arbitrary elements of M. Then*

$$\left| \Pr(h(x_j) = a_j, \textit{ for } 1 \le j \le d) - \frac{1}{m^d} \right| \le \frac{\epsilon(m, d, \ell, \delta)}{m^d},$$

where $\epsilon(m, d, \ell, \delta) = \frac{d}{\ell+1} \cdot (c \cdot d)^\ell \cdot m^{d-\ell\delta} = O(m^{d-\ell\delta})$.

Proof. We imagine that the random choices are made in two stages. First, the functions $f_1, \ldots, f_\ell$ are chosen, then the random group elements $m_{i,s}$. It is easy to see that if $f_1, \ldots, f_\ell$ are such that for each j, $2 \le j \le d$, we have

$$(A_j) \qquad \exists i, 1 \le i \le \ell \colon f_i(x_j) \notin \{f_i(x_1), \ldots, f_i(x_{j-1})\},$$

then for such an i the random element $m_{j,f_i(x_j)}$ will be independent of the sequence $(h(x_1), \ldots, h(x_{j-1}))$, hence $h(x_j)$ will also be independent of $(h(x_1), \ldots, h(x_{j-1}))$. Thus,

$$\Pr(h(x_j) = a_j, \text{ for } 1 \le j \le d \mid \forall j : (A_j)) = \frac{1}{m^d}.$$

It remains to show that the probability that (A_j) fails for some j is bounded by $\epsilon(m, d, \ell, \delta)/m^d$. This can be seen as follows:

By c-universality, for each i the probability that $f_i(x_j) \in \{f_i(x_1), \ldots, f_i(x_{j-1})\}$ is bounded by $(j-1)c/r$. By independence, $\Pr((A_j) \text{ fails}) \le ((j-1)c/r)^\ell$. Thus,

$$\begin{aligned} \Pr(\exists j \colon (A_j) \text{ fails}) &\le \sum_{2 \le j \le d} ((j-1)c/r)^\ell \\ &\le \frac{d^{\ell+1}}{\ell+1} \cdot \frac{c^\ell}{r^\ell} = \frac{d}{\ell+1} \cdot (d \cdot c)^\ell \cdot m^{d-\delta\ell} \cdot \frac{1}{m^d}. \end{aligned}$$

The last equation follows from $r = m^\delta$. □

5.2 One integer multiplication suffices

A simple variant of the construction described above makes it possible to implement approximately d-wise independent hash classes in such a way that for evaluating the function one integer multiplication as well as some simple bit operations and table lookups are sufficient. Consider the 2-universal class from [6] that consists of functions $f_a : \{0, 1, \ldots, 2^\kappa - 1\} \to \{0, 1, \ldots, 2^\mu - 1\}$ given by $f_a : x \mapsto (ax) \bmod 2^\kappa \operatorname{div} 2^{\kappa-\mu}$. We use the nice (and well-known) property of this class that one can evaluate ℓ functions $f_1, \ldots, f_\ell$ on one argument x by one multiplication of x with an $O(\ell\kappa)$-bit number and $O(\ell)$ bit level operations like shifts.

If we substitute this construction e. g. in the cuckoo hashing function, we see that we can run cuckoo hashing with functions that require only one multiplication (and bit operations and table lookups).

Acknowledgements

We thank Rasmus Pagh for many interesting discussions regarding cuckoo hashing and uniform hashing as well as for valuable comments on the manuscript. A large part of this work was done while the first author was visiting the Max-Planck-Institut für Informatik in Saarbrücken, Germany. The kind hospitality of the algorithms group there is gratefully acknowledged.

References

[1] B. Bollobás. *Random Graphs.* Academic Press, London, 1985.

[2] J. L. Carter and M . N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18:143–154, 1979.

[3] R. Diestel. *Graph Theory.* Springer-Verlag, New York, 1997.

[4] M. Dietzfelbinger. Universal hashing and k-wise independent random variables via integer arithmetic without primes. In *Proc. of 13th STACS*, vol. 1046 of *LNCS*, pp. 569–580, 1996.

[5] M. Dietzfelbinger, Y. Gil, Y. Matias, and N .Pippenger. Polynomial hash functions are reliable. In *Proc. of 19th ICALP*, vol. 623 of *LNCS*, pp. 235–246, 1992.

[6] M. Dietzfelbinger, T. Hagerup, J. Katajainen, and M. Penttonen. A reliable randomized algorithm for the closest-pair problem. *J. Algorithms*, 25:19–51, 1997.

[7] M. Dietzfelbinger and F. Meyer auf der Heide. Dynamic hashing in real time. In J. Buchmann, H. Ganzinger, and W. J. Paul, editors, *Informatik — Festschrift zum 60. Geburtstag von Günter Hotz*, pp. 95–119, 1992.

[8] M. Dietzfelbinger, A. Karlin, K. Mehlhorn, F. Meyer auf der Heide, H. Rohnert, and R. E. Tarjan. Dynamic perfect hashing: Upper and lower bounds. *SIAM J. Comput.* 23:738–761, 1994.

[9] E. A. Fox, L. S. Heath, Q. F. Chen, and A. M. Daoud. Practical minimal perfect hash functions for large databases. *Comm. ACM* 35(1):105–121, 1992.

[10] D. Fotakis, R. Pagh, P. Sanders, and P. Spirakis. Space efficient hash tables with worst case constant access time. In *Proc. of 20th STACS*, vol. 2607 of *LNCS*, pp. 271–282, 2003.

[11] R. M. Karp, M. Luby, and F. Meyer auf der Heide. Efficient PRAM simulation on a distributed memory machine. *Algorithmica*, 16:517–542, 1996.

[12] G. S. Lueker and M. Molodowitch. More analysis of double hashing. *Combinatorica*, 13:83–96, 1993.

[13] B. S. Majewski, N. C. Wormald, G. Havas, and Z. J. Czech. A family of perfect hashing methods. *The Computer Journal*, 39(6): 547–554, 1996.

[14] F. Meyer auf der Heide, C. Scheideler, and V. Stemann. Exploiting storage redundancy to speed up shared memory simulations. *Theoret. Comput. Sci.*, 162:245–281, 1996.

[15] A. Östlin and R. Pagh. Simulating uniform hashing in constant time and optimal space. Research report RS-02-27, BRICS, Department of Computer Science, University of Aarhus, 2002.

[16] A. Östlin and R. Pagh. Uniform hashing in constant time and linear space. In *Proc. of 35th ACM STOC (to appear)*, 2003.

[17] R. Pagh. Hash and displace: Efficient evaluation of minimal perfect hash functions. In *Proc. of 6th WADS*, vol. 1663 of *LNCS*, pp. 49–54, 1999.

[18] R. Pagh. On the cell probe complexity of membership and perfect hashing. In *Proc. of 33rd ACM STOC*, pp. 425–432, 2001.

[19] R. Pagh and F. F. Rodler. Cuckoo hashing. In *Proc. of 9th ESA*, vol. 2161 of *LNCS*, pp. 121–133, 2001. Expanded manuscript at `http://www.it-c.dk/people/pagh/papers.html`.

[20] J. P. Schmidt and A. Siegel. The analysis of closed hashing under limited randomness. In *Proc. of 22nd ACM STOC*, pp. 224–234, 1990.

[21] J. P. Schmidt and A. Siegel. Closed hashing is computable and optimally randomizable with universal hash functions. Technical Report, TR1995-687, Courant Institute, New York University, 1995.

[22] A. Siegel. On universal classes of fast high performance hash functions, their time-space trade-off, and their applications. In *Proc. of 30th IEEE FOCS*, pp. 20–25, 1989.

[23] A. Siegel. On universal classes of extremely random constant time hash functions and their time-space tradeoff. Technical Report TR1995-684, Courant Institute, New York University, April 1995.

[24] M. N. Wegman and J. L. Carter. New classes and applications of hash functions. In *Proc. of 20th IEEE FOCS*, pp. 175–182, 1979.

I N F O R M A T I K

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available by anonymous ftp from `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Anja Becker
Stuhlsatzenhausweg 85
66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-2003-NWG2-002	F. Eisenbrand	Fast integer programming in fixed dimension
MPI-I-2003-NWG2-001	L.S. Chandran, C.R. Subramanian	Girth and Treewidth
MPI-I-2003-4-001	M. Tarini, H.P.R. Lensch, M. Goesele, H. Seidel	3-D Aqusition of Mirrowing Objects
MPI-I-2003-2-001	P. Maier	Compositional Circular Assume-Guarantee Rules Cannot Be Sound And Complete
MPI-I-2003-1-010	H. Tamaki	Soon to be published
MPI-I-2003-1-009	B. Csaba	On the Bollobás – Eldridge conjecture for bipartite graphs
MPI-I-2003-1-008	P. Sanders	Soon to be published
MPI-I-2003-1-007	H. Tamaki	Alternating cycles contribution: a strategy of tour-merging for the traveling salesman problem
MPI-I-2003-1-006	H. Tamaki, M. Dietzfelbinger	On the probability of Rendezvous in Graph
MPI-I-2003-1-005	M. Dietzfelbinger, P. Woelfel	Almost Random Graphs with Simple Hash Functions
MPI-I-2003-1-004	E. Althaus, T. Polzin, S.V. Daneshmand	Improving Linear Programming Approaches for the Steiner Tree Problem
MPI-I-2003-1-003	R. Beier, B. Vcking	Random Knapsack in Expected Polynomial Time
MPI-I-2003-1-002	P. Krysta, P. Sanders, B. Vcking	Scheduling and Traffic Allocation for Tasks with Bounded Splittability
MPI-I-2003-1-001	P. Sanders, R. Dementiev	Asynchronous Parallel Disk Sorting
MPI-I-2002-4-002	F. Drago, W. Martens, K. Myszkowski, H. Seidel	Perceptual Evaluation of Tone Mapping Operators with Regard to Similarity and Preference
MPI-I-2002-4-001	M. Goesele, J. Kautz, J. Lang, H.P.A. Lensch, H. Seidel	Tutorial Notes ACM SM 02 A Framework for the Acquisition, Processing and Interactive Display of High Quality 3D Models
MPI-I-2002-2-008	W. Charatonik, J. Talbot	Atomic Set Constraints with Projection
MPI-I-2002-2-007	W. Charatonik, H. Ganzinger	Symposium on the Effectiveness of Logic in Computer Science in Honour of Moshe Vardi
MPI-I-2002-1-008	P. Sanders, J.L. Trff	The Factor Algorithm for All-to-all Communication on Clusters of SMP Nodes
MPI-I-2002-1-005	M. Hoefer	Performance of heuristic and approximation algorithms for the uncapacitated facility location problem
MPI-I-2002-1-004	S. Hert, T. Polzin, L. Kettner, G. Schfer	Exp Lab A Tool Set for Computational Experiments
MPI-I-2002-1-003	I. Katriel, P. Sanders, J.L. Trff	A Practical Minimum Scanning Tree Algorithm Using the Cycle Property

MPI-I-2002-1-002	F. Grandoni	Incrementally maintaining the number of l-cliques
MPI-I-2002-1-001	T. Polzin, S. Vahdati	Using (sub)graphs of small width for solving the Steiner problem
MPI-I-2001-4-005	H.P.A. Lensch, M. Goesele, H. Seidel	A Framework for the Acquisition, Processing and Interactive Display of High Quality 3D Models
MPI-I-2001-4-004	S.W. Choi, H. Seidel	Linear One-sided Stability of MAT for Weakly Injective Domain
MPI-I-2001-4-003	K. Daubert, W. Heidrich, J. Kautz, J. Dischler, H. Seidel	Efficient Light Transport Using Precomputed Visibility
MPI-I-2001-4-002	H.P.A. Lensch, J. Kautz, M. Goesele, H. Seidel	A Framework for the Acquisition, Processing, Transmission, and Interactive Display of High Quality 3D Models on the Web
MPI-I-2001-4-001	H.P.A. Lensch, J. Kautz, M. Goesele, W. Heidrich, H. Seidel	Image-Based Reconstruction of Spatially Varying Materials
MPI-I-2001-2-006	H. Nivelle, S. Schulz	Proceeding of the Second International Workshop of the Implementation of Logics
MPI-I-2001-2-005	V. Sofronie-Stokkermans	Resolution-based decision procedures for the universal theory of some classes of distributive lattices with operators
MPI-I-2001-2-004	H. de Nivelle	Translation of Resolution Proofs into Higher Order Natural Deduction using Type Theory
MPI-I-2001-2-003	S. Vorobyov	Experiments with Iterative Improvement Algorithms on Completely Unimodel Hypercubes
MPI-I-2001-2-002	P. Maier	A Set-Theoretic Framework for Assume-Guarantee Reasoning
MPI-I-2001-2-001	U. Waldmann	Superposition and Chaining for Totally Ordered Divisible Abelian Groups
MPI-I-2001-1-007	T. Polzin, S. Vahdati	Extending Reduction Techniques for the Steiner Tree Problem: A Combination of Alternative-and Bound-Based Approaches
MPI-I-2001-1-006	T. Polzin, S. Vahdati	Partitioning Techniques for the Steiner Problem
MPI-I-2001-1-005	T. Polzin, S. Vahdati	On Steiner Trees and Minimum Spanning Trees in Hypergraphs
MPI-I-2001-1-004	S. Hert, M. Hoffmann, L. Kettner, S. Pion, M. Seel	An Adaptable and Extensible Geometry Kernel
MPI-I-2001-1-003	M. Seel	Implementation of Planar Nef Polyhedra
MPI-I-2001-1-002	U. Meyer	Directed Single-Source Shortest-Paths in Linear Average-Case Time
MPI-I-2001-1-001	P. Krysta	Approximating Minimum Size 1,2-Connected Networks
MPI-I-2000-4-003	S.W. Choi, H. Seidel	Hyperbolic Hausdorff Distance for Medial Axis Transform
MPI-I-2000-4-002	L.P. Kobbelt, S. Bischoff, K. Khler, R. Schneider, M. Botsch, C. Rssl, J. Vorsatz	Geometric Modeling Based on Polygonal Meshes
MPI-I-2000-4-001	J. Kautz, W. Heidrich, K. Daubert	Bump Map Shadows for OpenGL Rendering
MPI-I-2000-2-001	F. Eisenbrand	Short Vectors of Planar Lattices Via Continued Fractions
MPI-I-2000-1-005	M. Seel, K. Mehlhorn	Infimaximal Frames: A Technique for Making Lines Look Like Segments
MPI-I-2000-1-004	K. Mehlhorn, S. Schirra	Generalized and improved constructive separation bound for real algebraic expressions
MPI-I-2000-1-003	P. Fatourou	Low-Contention Depth-First Scheduling of Parallel Computations with Synchronization Variables
MPI-I-2000-1-002	R. Beier, J. Sibeyn	A Powerful Heuristic for Telephone Gossiping
MPI-I-2000-1-001	E. Althaus, O. Kohlbacher, H. Lenhof, P. Mller	A branch and cut algorithm for the optimal solution of the side-chain placement problem
MPI-I-1999-4-001	J. Haber, H. Seidel	A Framework for Evaluating the Quality of Lossy Image Compression

MPI-I-1999-3-005	T.A. Henzinger, J. Raskin, P. Schobbens	Axioms for Real-Time Logics
MPI-I-1999-3-004	J. Raskin, P. Schobbens	Proving a conjecture of Andreka on temporal logic
MPI-I-1999-3-003	T.A. Henzinger, J. Raskin, P. Schobbens	Fully Decidable Logics, Automata and Classical Theories for Defining Regular Real-Time Languages