RWTH Aachen

Department of Computer Science
Technical Report

Derandomizing Non-uniform Color-Coding I

Joachim Kneis, Alexander Langer, Peter Rossmanith

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2009-07

RWTH Aachen · Department of Computer Science · March 2009

Derandomizing Non-uniform Color-Coding I*

Joachim Kneis, Alexander Langer, Peter Rossmanith

Theoretical Computer Science Group
RWTH Aachen University, Germany
Email: {kneis,langer,rossmani}@cs.rwth-aachen.de

Abstract. Color-coding, as introduced by Alon, Yuster, and Zwick, is a well-known tool for algorithm design and can often be efficiently derandomized using universal hash functions. In the special case of only two colors, one can use (n,k)-universal sets for the derandomization. In this paper, we introduce (n,k,l)-universal sets that are typically smaller and can be constructed faster. Nevertheless, for some problems they are still sufficient for derandomization and faster deterministic algorithms can be obtained. This particularly works well when the color-coding does not use a uniform probability distribution. To exemplify the concept, we present an algorithm for the Unique Coverage problem introduced by Demaine, Feige, Hajiaghayi, and Salavatipour. The example also shows how to extend the concept to multiple colors.

1 Introduction

The principle of randomization is well-established in the field of algorithm design. Due to the sometimes non-intuitive, even surprising laws of probability theory (cf., the *birthday paradoxon*), randomized algorithms often can find correct solutions faster than any known deterministic algorithm.

Consider, for example, the popular 3-SAT problem, which can be solved in time bounded by $O^*(1.324^n)$ using a randomized algorithm with constant error probability [10, 17], but only in time bounded by $O^*(1.473^n)$ when a purely deterministic approach is used [6, 2].

Even worse, it can be shown that there are problems, for which *every* deterministic approach has a certain worst-case lower bound, while a randomized algorithm typically runs much faster. Suppose, for example, that there are 100 boxes and exactly half of them contain a present. How long does it take to find one? Every deterministic algorithm can be forced to look into at least 51 boxes before finding a present. However, a randomized algorithm repeatedly choosing an arbitrary box with uniform probability 1/100 until finding a present is expected to guess only two times. Here, the worst- and average-case coincide, which explains the large gap between deterministic and randomized bounds.

Similarly, there are examples that are not so naïve, yet surprising: Let's say, we are are given an array of n integers, and we know that either the array is sorted, or it is not sorted and you have to delete at least $n/10$ elements to obtain a sorted array. The goal is to design an algorithm that decides which of the two cases holds. Intuitively, one could argue you always have to compare all pairs of subsequent elements, and in fact deterministically $\Omega(n)$ elements have to be examined. Surprisingly, there is a randomized algorithm that runs in $O(\log n)$ time [8].

* Supported by the DFG under grant RO 927/8

Randomized algorithms are algorithms that have access to true random bits. They can be classified into *Monte Carlo* and *Las Vegas* algorithms: Monte Carlo algorithms have a fixed running time, but the outcome has to be correct with a probability of at least, say, 2/3. Las Vegas algorithms always return the correct result, but their running time depends on the random bits. In this paper, we consider Monte Carlo algorithms.

Often you can turn a randomized algorithm into a deterministic one by techniques commonly referred to as *derandomization.* One possibility is to repeatedly feed the algorithm several bit strings of length n instead of n random bits. Usually, however, a randomized algorithm requires random bits that are independent, i.e., for n random bits, every combination occurs with equal probability 2^{-n}. One possibility to derandomize such an algorithm is therefore to run it 2^n times with every possible bit string of length n, which is usually too slow to be of practical interest.

Sometimes, though, algorithms do not require n independent random bits. Sometimes, it suffices if only every subset of $k < n$ bits is independent. A set of n-bit vectors with this property is called *(n, k)-independent.* Such randomized algorithms that only require (n, k)-independent bits can often be derandomized with the help of *(n, k)-universal sets.*

Definition 1. *Let n and k be integers. An (n, k)-universal set is a set Ω of bit strings $b = b_0 \dots b_{n-1} \in \{0, 1\}^n$, such that for all distinct positions $i_1, \dots, i_k \in Z_n$ and all patterns $x = x_1 \dots x_k \in \{0, 1\}^k$ there is some $b = b_0 \dots b_{n-1} \in \Omega$ with $b_{i_j} = x_j$ for all $1 \le j \le k$.*

Naor, Schulman, and Srinivasan [15] showed how to construct (n, k)-universal sets of size $2^{k+O(\log^2 k)} \log n$ in linear time. This is nearly optimal, because there is a lower bound of $\Omega(2^k \log n)$ on the size of (n, k)-universal sets [11].

Algorithms that can often be derandomized using (n, k)-universal sets are those that use the so called *color-coding* technique [1], the *random separation* technique [3], and those that use the *randomized divide-and-conquer* (also called *divide-and-color*) [4, 5, 12] approach. In Section 2.1, we illustrate the random separation technique in further detail on the problem Exact Partial Vertex Cover: Given a graph and two integers k and t, is there a set C of k nodes adjacent to exactly t edges? In short, the algorithm randomly partitions the nodes into two sets colored 0 and 1, and then uses dynamic programming to find a solution C that is colored with color 1 and whose neighborhood is colored with color 0. This coloring step succeeds with a probability of at least 2^{-k-t} when uniform probabilities for the two colors are used.[1] Furthermore, (n, k)-universal sets can be used to obtain a deterministic algorithm with a running time of $2^{k+t+O(\log^2(k+t))} poly(n)$.

Sometimes, however, it is not optimal to choose the colors with uniform probabilities. This is, for example, the case, when much fewer elements shall receive the color 0 than shall receive the color 1. Assume, for instance, that an

[1] Note that a randomized algorithm with running time $t(n)$ and success probability 2^{-x} can easily be turned into a randomized algorithm with constant success probability arbitrarily close to one and a running time of $O(2^x t(n))$. This standard technique is called *probability amplification* (see, e.g., [14]). Whenever we compare randomized algorithms with deterministic algorithms, we mean the randomized algorithm with constant success probability.

instance of EXACT PARTIAL VERTEX COVER is such that $t = k^2$, and therefore k^2 nodes shall be colored with 0, but only k nodes with 1. Then the randomized algorithm's success probability is only 2^{-k^2-k} when using uniform probabilities, but can be improved to

$$(1/k)^k(1-1/k)^{k^2} \sim e^{-k\ln k - k + 1/2}$$

by using probabilities $1/k$ for the color 1 and $1-1/k$ for the color 0. See Section 2.2 for details. If we now simply use (n, k^2+k)-universal sets for the derandomization, we still only obtain a deterministic algorithm with running time of $2^{k^2+k+O(\log k)}poly(n)$, which is much slower than the corresponding randomized algorithm.

Please note, however, that here we do not need the whole power of (n, k^2+k)-universal sets: We do not need to find all possible subpatterns, but only those having k ones. To capture this concept, we introduce (n, k, l)-*universal sets*[2]:

Definition 2. *Let $n \geq k \geq l$ be integers. An (n, k, l)-universal set is a set Ω of bit strings $b = b_0 \dots b_{n-1} \in \{0,1\}^n$, such that for all distinct positions $i_1, \dots, i_k \in Z_n$ and all patterns $x = x_1 \dots x_k \in \{0,1\}^k$ with Hamming weight at most l, there is some $b = b_0 \dots b_{n-1} \in \Omega$ with $b_{i_j} = x_j$ for all $1 \leq j \leq k$.*

Using (n, k^2+k, k)-universal sets, we can get a better deterministic time bound if we are able to construct (n, k^2+k, k)-universal sets much faster than (n, k^2+k)-universal sets. The main result of this paper are (n, k, l)-universal sets of size $2nk^{2l}$ that can be constructed in time $O(n^2k^{2l+2})$.

1.1 Outline

In this paper, we present a construction of (n, k, l)-universal sets, which is self-contained, very simple, and can easily be implemented, but has two drawbacks:

1. The size of the sets is not the smallest possible.
2. The method works well for $l \ll k$, but is not optimal for $l = \Theta(k)$, in particular for $l = k/2$.

These issues will be addressed in the second part, where we will present a more complicated technique, which allows for better constants and is useful for all combinations of k and l.

This paper is organized as follows: In Section 2.1 we introduce the concept of *random separation* with uniform probabilities using the example of the EXACT PARTIAL VERTEX COVER problem. In Section 2.2, we show how the running time can be improved when certain conditions are met and motivate the introduction of (n, k, l)-universal sets. Our construction of (n, k, l)-universal sets is presented in Section 3. Another application is shown in Section 4, which contains the currently fastest parameterized algorithm for the UNIQUE COVERAGE problem introduced by Demaine, Feige, Hajiaghayi, and Salavatipour [7].

[2] Please do not confuse those with (n, k, l)-splitters [15], which are k-universal hash functions on l colors, i.e., a $(n, k, 2)$-splitter is also an (n, k)-universal set.

2 Random Separation

The *random separation* technique has been introduced recently by Cai, Chan, and Chan [3]. In this section, we want to illustrate their technique using the problem Exact Partial Vertex Cover, which is defined as follows.

Exact Partial Vertex Cover
Input: An undirected graph $G = (V, E)$, positive integers k, t
Question: Is there a set $C \subseteq V$, $|C| \leq k$, such that exactly t edges are adjacent to a node in C?

We also say that such a set C *covers* t edges and call C a *partial vertex cover* or *t-vertex cover*. The basic idea of the random separation technique is to randomly assign colors 0 and 1 to the nodes in V, such that a solution can be built from 1-colored components in G. If a solution exists at all, it can be shown that a random coloring is successful with a certain probability (here, 2^{-k-t}). This already determines the success probability of the whole algorithm.

Therefore, let $c\colon V \to \{0, 1\}$ be a mapping (called a *coloring*). A set of nodes $U \subseteq V$ is called a *c-1-component* if U is a connected component of $G[c^{-1}(1)]$.

2.1 Coloring with uniform probabilities

We first study the general case, where we use uniform probabilities for the two colors.

Lemma 1 (Cai, Chan, Chan [3]). *Let $G = (V, E)$, and let $C \subseteq V$ be a t-vertex-cover of size k. If a coloring $c\colon V \to \{0, 1\}$ is chosen randomly, such that each node is colored 1 or 0 independently with uniform probability 1/2, then with probability at least 2^{-k-t}, C consists of c-1-components.*

Proof. C contains k nodes, and since C is a t-vertex cover, exactly t edges are adjacent to C. In particular, there are at most t nodes in $N(C) := \bigcup_{v \in C} N(v) \setminus C$. Thus, with probability 2^{-k} all k nodes in C are colored 1, and with probability at least 2^{-t} all nodes in $N(C)$ are colored 0. Components of $G[C]$ are therefore c-1-components of G with probability at least 2^{-k-t}. □

Once a coloring c has been chosen, further computation is required to find c-1-components that actually yield a t-vertex cover. This computation can easily be done in polynomial time using the standard dynamic programming approach for the classical Subset Sum problem. For further details, please see [3].

Note that the proof does not require that all nodes, but only that up to $k + t$ of them are colored independently. Therefore, the above algorithm can easily be derandomized using $(n, k + t)$-universal sets. The corresponding deterministic algorithm then has a running time bounded by

$$2^{k+t+O(\log^2(k+t))} poly(n).$$

2.2 Coloring with non-uniform probabilities

Now assume that the instance of Exact Partial Vertex Cover is such that $t = k^2$. Without modifications, the above randomized approach then yields a success probability of only 2^{-k-k^2}. However, the probability can easily be improved by choosing non-uniform probabilities for 0 and 1:

Lemma 2. *Let $G = (V, E)$, and let $C \subseteq V$ be a t-vertex-cover of size k. If a coloring $c\colon V \to \{0,1\}$ is chosen randomly, such that each node is colored independently and receives the color 1 with probability $1/k$ and the color 0 with probability $1 - 1/k$, then C can be partitioned into c-1-components with a probability of at least $k^{-k}e^{-k+1/2}(1 + O(k^{-1})) = e^{-k\ln k - k + 1/2}(1 + O(k^{-1}))$.*

Proof. First note, that by the Taylor expansion of $\ln(1 + z)$,

$$k^2 \ln\Bigl(1 - \frac{1}{k}\Bigr) = k^2\Bigl(-\frac{1}{k} + \frac{1}{2k^2} + O(k^{-3})\Bigr) = -k + \frac{1}{2} + O(k^{-1}). \tag{1}$$

Now, again k nodes in C must be colored 1 and up to $t = k^2$ nodes in $N(C)$ must be colored 0. Hence, the probability that C and $N(C)$ are colored correctly is at least

$$\begin{aligned}\Bigl(\frac{1}{k}\Bigr)^k\Bigl(1 - \frac{1}{k}\Bigr)^{k^2} &= k^{-k}\exp\Bigl[k^2\ln\Bigl(1 - \frac{1}{k}\Bigr)\Bigr] \\ &\overset{(1)}{=} k^{-k}\exp\Bigl[-k + 1/2 + O\Bigl(\frac{1}{k}\Bigr)\Bigr] = e^{-k\ln k - k + 1/2}\Bigl[1 + O\Bigl(\frac{1}{k}\Bigr)\Bigr].\end{aligned} \tag{2}$$

□

Unfortunately, if we derandomize this algorithm using $(n, k^2 + k)$-universal sets, we obtain a running time of

$$2^{k+k^2+O(\log^2(k+k^2))}poly(n) = 2^{k+k^2+O(\log^2 k)}poly(n), \tag{3}$$

which is much slower than its corresponding randomized algorithm.

However, we can use (n, k^2+k, k)-universal sets instead, because only k nodes must be colored with color 1. In the next section, we will introduce (n, k, l)-universal sets for arbitrary $l \le k$ that can be constructed in time $O(n^2k^{2l+2})$. Therefore, the randomized algorithm using non-uniform probabilities can be derandomized faster and a deterministic running time of

$$\begin{aligned}(k^2 + k)^{2k+2}poly(n) &= k^{4k+4}(1 + 1/k)^{2k+2}poly(n) \\ &= k^{4k+4}poly(n) = e^{4k\ln k}poly(n)\end{aligned}$$

can be achieved (here, note that $(1 + 1/k)^k \le e$ and k^4 is hidden in $poly(n)$), which is dramatically faster than (3).

3 A simple construction of (n, k, l)-universal sets

In this section, we show how to construct (n, k, l)-universal sets. The style of presentation is inspired by [4].

Let $Z_n := \{0, \ldots, n-1\}$. For all $n, k, z \in \mathbf{N}$, let

$$g_{n,k,z}\colon Z_n \to Z_{k^2},\ a \mapsto (az \bmod q_0) \bmod k^2,$$

where q_0 is the smallest prime larger than or equal to n.

Proposition 1 (Fredman, Komlos, and Szemerédi [9]).
Let $n \le k \le l$ be integers and let q_0 be the smallest prime larger than or equal to n. Then, for all $S \subseteq Z_n$ with $|S| \le k$ there is $z \le q_0$ such that $g_{n,k,z}$ restricted to S is injective.

Lemma 3. *Let $n \geq k \geq l$ be integers. There is an (n, k, l)-universal set of size at most n^l that can be constructed in time $O(n^{l+1})$.*

Proof. There are $\binom{n}{i}$ many possibilities to place i ones into a bit string of length n and therefore

$$\sum_{i=0}^{l} \binom{n}{i} \leq n^l$$

possibilities to place up to i ones into such a bit string. □

Theorem 1. *Let $n \geq k \geq l$ be integers. There is an (n, k, l)-universal set of size at most $2nk^{2l}$ that can be constructed in time $O(n^2 k^{2l+2})$.*

Proof. By the Bertrand-Chebyshev Theorem (see, e.g., [16]), there is a prime q_0 with $n \leq q_0 \leq 2n$. Let Ω' be a (k^2, k, l)-universal set. We define our (n, k, l)-universal set Ω as follows:

$$\Omega := \left\{ b_{g_{n,k,z}(0)} \ldots b_{g_{n,k,z}(n-1)} \;\middle|\; 0 \leq z \leq q_0, b_0 \ldots b_{k^2-1} \in \Omega' \right\}$$

Since $q_0 \leq 2n$ and, by Lemma 3, $|\Omega'| \leq k^{2l}$, we easily conclude $|\Omega| \leq 2nk^{2l}$. To see that Ω is (n, k, l)-universal, let $x = x_1 \ldots x_k \in \{0, 1\}^k$ such that $\sum_{i=1}^{k} x_i \leq l$ be a bit string and let $i_1, \ldots, i_k \in Z_n$ be positions. We have to show that Ω contains a bit string $c_0 \ldots c_{n-1}$ with $c_{i_j} = x_j$ for all $1 \leq j \leq k$. By Proposition 1, we can choose a $z \leq q_0$ such that $g_{n,k,z} \colon Z_n \to Z_{k^2}$ is injective when restricted to the set $\{i_1, \ldots, i_k\}$. Let $i'_j := g_{n,k,z}(i_j)$. Because it is (k^2, k, l)-universal, Ω' contains some $b \in \{0, 1\}^{k^2}$ with $b_{i'_j} = x_j$ for all $1 \leq j \leq k$. Since $c_{i_j} = b_{g_{n,k,z}(i_j)} = b_{i'_j} = x_j$, Ω is (n, k, l)-universal.

The desired running time follows from the size of Ω', $0 \leq z \leq q_0 \leq 2n$, and each $\omega \in \Omega$ having length n. □

Corollary 1. *Let $n \geq k$ and d be integers. There is an (n, k^d, k)-universal set of size at most $2nk^{2dk}$ that can be constructed in time $O(n^2 k^{2(k+1)d})$.*

4 Unique Coverage

The Unique Coverage problem was introduced by Demaine, Feige, Hajiaghayi, and Salavatipour [7] and is defined as follows:

Unique Coverage
Input: A universe U, a collection $\mathcal{S} \subseteq 2^U$ of subsets of U, and $k \in \mathbf{N}$
Parameter: k
Question: Is there a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ that covers at least k elements from U *uniquely*, i.e., each occurs in exactly one set of $\mathcal{S}'$.

Moser, Raman, and Sikdar [13] showed that Unique Coverage is fixed parameter tractable by bounding the size of $\mathcal{S}$ by 4^k. Since testing subfamilies of $\mathcal{S}$ that contain up to k elements is sufficient, the running time required to solve the problem is bounded by $O(4^{k^2} \mathit{poly}(n))$. Using color-coding and derandomization, we can improve the bound as follows: We first prove that Algorithm 1, which uses non-uniform color-coding, has a success probability of at least $e^{-2k \ln k - k + 1/2}(1 + O(k^{-1}))$. We then show how to efficiently derandomize the color-coding using

Algorithm 1 Solving UNIQUE COVERAGE on inputs U, $\mathcal{S} \in 2^U$ and $k \in \mathbf{N}$

0. If $\mathcal{S}$ contains a set S with $|S| \geq k$, answer "yes".
1. For all $u \in U$ (independently) assign u the color A with probability $1/k$ and the color 0 with probability $1 - 1/k$.
2. Delete all $u \in U$ colored with color 0.
3. For all remaining $u \in U$, color u uniformly and independently with the colors $1, \ldots, k$.
4. Delete all $S \in \mathcal{S}$ that contain two elements of different colors.
5. For all $i = 1, \ldots, k$, choose an S^i of maximal cardinality among all remaining $S \in \mathcal{S}$ that contain elements of color i. If no such set exists, let $S^i := \emptyset$.
6. If $|S^1| + \cdots + |S^k| \geq k$, then answer "yes" else answer "no".

(n, k, l)-universal sets. This yields a deterministic algorithm deciding UNIQUE COVERAGE within a run time bound of $O(k^{4k} c^k \mathit{poly}(n))$.

We first explain how the algorithm works: In step 1, a first color-coding assigns the color A to those elements that are possibly covered uniquely in some solution. Elements, that are not of interest in some solution are supposed to receive the color 0, and since they do not contribute to the solution further, we delete them afterwards. In step 3, a second color-coding then colors all remaining elements with a unique color i indicating the set S^i that is supposed to cover them uniquely. In a final step, it is then sufficient to pick the sets with the highest cardinality among those of the same color.

Theorem 2. *The* UNIQUE COVERAGE *problem can be solved by a polynomial time, randomized algorithm with a success probability of at least* $e^{-2k \ln k - k + 1/2}(1 + O(k^{-1}))$.

Proof. Without loss of generality, we can assume that $|S| < k$ for all $S \in \mathcal{S}$. Let $\mathcal{S}' \subseteq \mathcal{S}$ be a solution that covers the elements $u_1, \ldots, u_k$ uniquely. Without loss of generality, we may furthermore assume each $S \in \mathcal{S}'$ covers at least one element uniquely, and hence $\mathcal{S}'$ contains at most k sets. This implies that $U_{\mathcal{S}'} := \bigcup_{S \in \mathcal{S}'} S$ contains less than k^2 elements. Now let $S^1, \ldots, S^l$ be the $l \leq k$ sets that cover the elements $u_1, \ldots, u_k$ uniquely.

It is easy to see that Algorithm 1 answers "yes" when in step 1 the elements $u_1, \ldots, u_k$ are being assigned the color A but all the elements in $U_{\mathcal{S}'} \setminus \{u_1, \ldots, u_k\}$ are colored with 0, and in the subsequent step 3 each u_i is being assigned the color j of its respective S^j it is covered by. It is therefore sufficient to estimate the probability of these two events, which then determines the success probability of Algorithm 1.

Hence, let X be the event that the algorithm answers "yes", let X_{0A} be the event that the algorithm correctly colors each of the (less than k^2) elements in $U_{\mathcal{S}'}$ with colors 0 and A, and let finally be $X_{1 \cdots k}$ be the event that each u_i receives the color j of its container S^j.

We easily obtain $\Pr(X_{1 \cdots k} \mid X_{0A}) \geq (1/k)^k = k^{-k}$ and

$$\begin{aligned}\Pr(X_{0A}) &\geq (1/k)^k (1 - 1/k)^{k^2 - k} \geq (1/k)^k (1 - 1/k)^{k^2} \\ &\overset{(2)}{=} e^{-k \ln k - k + 1/2} \Big[1 + O\Big(\frac{1}{k}\Big) \Big].\end{aligned}$$

Therefore, the overall success probability is

$$\begin{aligned}\Pr(X) &= \Pr(X_{0A}) \cdot \Pr(X_{1\cdots k} \mid X_{0A}) \\ &\geq e^{-k\ln k - k + 1/2}\Big[1 + O\Big(\frac{1}{k}\Big)\Big] \cdot k^{-k} \\ &= e^{-2k\ln k - k + 1/2}\Big[1 + O\Big(\frac{1}{k}\Big)\Big].\end{aligned}$$

□

We immediately conclude:

Theorem 3. *There is a deterministic algorithm deciding the* Unique Coverage *problem in time bounded by $O(k^{4k}c^k poly(n))$ (c being some constant).*

Proof. Note that only the k elements $u_1, \ldots, u_k$ must be colored with A, the remaining $k^2 - k$ elements are to be colored by 0. This calls for an (n, k^2, k)-universal set Ω as introduced in the previous section. For each coloring $\omega \in \Omega$ we can then use k-universal hash functions [1] to derandomize the second random coloring, which can be done in time $c^k \log n$ for some constant c. By Theorem 1, the running time of the derandomized algorithm is therefore bounded by $O(k^{4k}c^k poly(n))$ (again, k^4 is hidden in $poly(n)$). □

It is easy to see that a variant of Algorithm 1, which uses the classic pseudo-polynomial Knapsack dynamic programming algorithm in steps 5 and 6, solves the following weighted version of the Unique Coverage within the same time bounds.

Budgeted Unique Coverage [7]

Input:	A universe U, a collection $\mathcal{S} \subseteq 2^U$, a cost function $c\colon \mathcal{S} \to \mathbf{N}$, a profit function $p\colon U \to \mathbf{N}$, integers B and P
Parameter:	P
Question:	Is there a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ that covers elements $u_1, \ldots, u_k$ uniquely, such that $c(\mathcal{S}') \leq B$ and $p(u_1) + \cdots + p(u_k) \geq P$?

References

1. N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995.
2. T. Brüggemann and W. Kern. An improved deterministic local search algorithm for 3-sat. *Theoretical Computer Science*, 329(1-3):303–313, 2004.
3. L. Cai, S. M. Chan, and S. O. Chan. Random separation: A new method for solving fixed-cardinality optimization problems. In *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC)*, number 4169 in Lecture Notes in Computer Science, pages 239–250. Springer-Verlag, 2006.
4. J. Chen, J. Kneis, S. Lu, D. Mölle, S. Richter, P. Rossmanith, S. Sze, and F. Zhang. Randomized divide-and-conquer: Improved path, matching, and packing algorithms. *SIAM Journal on Computing*, 2009. to appear.
5. J. Chen, S. Lu, S.-H. Sze, and F. Zhang. Improved algorithms for path, matching, and packing problems. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 298–307, 2007.
6. E. Dantsin, A. Goerdt, E. A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, and U. Schöning. A deterministic $(2 - 2/(k+1))^n$ algorithm for k-sat based on local search. *Theoretical Computer Science*, 289(1):69–83, 2002.
7. E. D. Demaine, U. Feige, M. T. Hajiaghayi, and M. R. Salavatipour. Combination can be hard: Approximability of the unique coverage problem. *SIAM Journal on Computing*, 2009. to appear.

8. F. Ergun, S. Kannan, S. R. Kumar, R. Rubinfeld, and M. Vishwanthan. Spot-checkers. *Journal of Computer and System Sciences*, 60:717–751, 2000.
9. M. Fredman, J. Komlos, and E. Szemerédi. Storing a sparse table with $O(1)$ worst case access time. *Journal of the ACM*, 31:538–544, 1984.
10. K. Iwama and S. Tamaki. Improved upper bounds for 3-SAT. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 328–328, 2004.
11. D. J. Kleitman and J. Spencer. Families of k-independent sets. *Discrete Mathematics*, 6:255–262, 1973.
12. J. Kneis, D. Mölle, S. Richter, and P. Rossmanith. Divide-and-color. In *Proceedings of the 32nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, number 4271 in Lecture Notes in Computer Science, pages 58–67. Springer-Verlag, 2006.
13. H. Moser, V. Raman, and S. Sikdar. The parameterized complexity of the unique coverage problem. In *Proceedings of the 18th International Symposium on Algorithms and Computation (ISAAC)*, number 4835 in Lecture Notes in Computer Science, pages 621–631. Springer-Verlag, 2007.
14. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
15. M. Naor, L. J. Schulman, and A. Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 182–190, 1995.
16. S. Ramanujan. A proof of Bertrand's Postulate. *Journal of the Indian Mathematical Society*, XI:181–182, 1919.
17. U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 410–414, 1999.

Aachener Informatik-Berichte

This list contains all technical reports published during the past five years. A complete list of reports dating back to 1987 is available from http://aib.informatik.rwth-aachen.de/. To obtain copies consult the above URL or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

2004-01 *	Fachgruppe Informatik: Jahresbericht 2003
2004-02	Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic
2004-03	Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting
2004-04	Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming
2004-05	Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming
2004-06	Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming
2004-07	Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination
2004-08	Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information
2004-09	Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity
2004-10	Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules
2005-01 *	Fachgruppe Informatik: Jahresbericht 2004
2005-02	Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"
2005-03	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions
2005-04	Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem
2005-05	Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots
2005-06	Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information
2005-07	Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks
2005-08	Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut
2005-09	Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures
2005-10	Benedikt Bollig: Automata and Logics for Message Sequence Charts
2005-11	Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture

2005-12 Neeraj Mittal, Felix Freiling, S. Venkatesan, Lucia Draque Penso: Efficient Reductions for Wait-Free Termination Detection in Faulty Distributed Systems

2005-13 Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments

2005-14 Felix C. Freiling, Sukumar Ghosh: Code Stabilization

2005-15 Uwe Naumann: The Complexity of Derivative Computation

2005-16 Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)

2005-17 Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)

2005-18 Thomas von der Maßen, Klaus Müller, John MacGregor, Eva Geisberger, Jörg Dörr, Frank Houdek, Harbhajan Singh, Holger Wußmann, Hans-Veit Bacher, Barbara Paech: Einsatz von Features im Software-Entwicklungsprozess - Abschlußbericht des GI-Arbeitskreises "Features"

2005-19 Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers

2005-20 Felix C. Freiling, Martin Mink: Bericht über den Workshop zur Ausbildung im Bereich IT-Sicherheit Hochschulausbildung, berufliche Weiterbildung, Zertifizierung von Ausbildungsangeboten am 11. und 12. August 2005 in Köln organisiert von RWTH Aachen in Kooperation mit BITKOM, BSI, DLR und Gesellschaft fuer Informatik (GI) e.V.

2005-21 Thomas Noll, Stefan Rieger: Optimization of Straight-Line Code Revisited

2005-22 Felix Freiling, Maurice Herlihy, Lucia Draque Penso: Optimal Randomized Fair Exchange with Secret Shared Coins

2005-23 Heiner Ackermann, Alantha Newman, Heiko Röglin, Berthold Vöcking: Decision Making Based on Approximate and Smoothed Pareto Curves

2005-24 Alexander Becher, Zinaida Benenson, Maximillian Dornseif: Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks

2006-01 * Fachgruppe Informatik: Jahresbericht 2005

2006-02 Michael Weber: Parallel Algorithms for Verification of Large Systems

2006-03 Michael Maier, Uwe Naumann: Intraprocedural Adjoint Code Generated by the Differentiation-Enabled NAGWare Fortran Compiler

2006-04 Ebadollah Varnik, Uwe Naumann, Andrew Lyons: Toward Low Static Memory Jacobian Accumulation

2006-05 Uwe Naumann, Jean Utke, Patrick Heimbach, Chris Hill, Derya Ozyurt, Carl Wunsch, Mike Fagan, Nathan Tallent, Michelle Strout: Adjoint Code by Source Transformation with OpenAD/F

2006-06 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Divide-and-Color

2006-07 Thomas Colcombet, Christof Löding: Transforming structures by set interpretations

2006-08 Uwe Naumann, Yuxiao Hu: Optimal Vertex Elimination in Single-Expression-Use Graphs

2006-09 Tingting Han, Joost-Pieter Katoen: Counterexamples in Probabilistic Model Checking

2006-10 Mesut Günes, Alexander Zimmermann, Martin Wenig, Jan Ritzerfeld, Ulrich Meis: From Simulations to Testbeds - Architecture of the Hybrid MCG-Mesh Testbed

2006-11 Bastian Schlich, Michael Rohrbach, Michael Weber, Stefan Kowalewski: Model Checking Software for Microcontrollers

2006-12 Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker: Replaying Play in and Play out: Synthesis of Design Models from Scenarios by Learning

2006-13 Wong Karianto, Christof Löding: Unranked Tree Automata with Sibling Equalities and Disequalities

2006-14 Danilo Beuche, Andreas Birk, Heinrich Dreier, Andreas Fleischmann, Heidi Galle, Gerald Heller, Dirk Janzen, Isabel John, Ramin Tavakoli Kolagari, Thomas von der Maßen, Andreas Wolfram: Report of the GI Work Group "Requirements Management Tools for Product Line Engineering"

2006-15 Sebastian Ullrich, Jakob T. Valvoda, Torsten Kuhlen: Utilizing optical sensors from mice for new input devices

2006-16 Rafael Ballagas, Jan Borchers: Selexels: a Conceptual Framework for Pointing Devices with Low Expressiveness

2006-17 Eric Lee, Henning Kiel, Jan Borchers: Scrolling Through Time: Improving Interfaces for Searching and Navigating Continuous Audio Timelines

2007-01 * Fachgruppe Informatik: Jahresbericht 2006

2007-02 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl: SAT Solving for Termination Analysis with Polynomial Interpretations

2007-03 Jürgen Giesl, René Thiemann, Stephan Swiderski, and Peter Schneider-Kamp: Proving Termination by Bounded Increase

2007-04 Jan Buchholz, Eric Lee, Jonathan Klein, and Jan Borchers: coJIVE: A System to Support Collaborative Jazz Improvisation

2007-05 Uwe Naumann: On Optimal DAG Reversal

2007-06 Joost-Pieter Katoen, Thomas Noll, and Stefan Rieger: Verifying Concurrent List-Manipulating Programs by LTL Model Checking

2007-07 Alexander Nyßen, Horst Lichter: MeDUSA - MethoD for UML2-based Design of Embedded Software Applications

2007-08 Falk Salewski and Stefan Kowalewski: Achieving Highly Reliable Embedded Software: An empirical evaluation of different approaches

2007-09 Tina Kraußer, Heiko Mantel, and Henning Sudbrock: A Probabilistic Justification of the Combining Calculus under the Uniform Scheduler Assumption

2007-10 Martin Neuhäußer, Joost-Pieter Katoen: Bisimulation and Logical Preservation for Continuous-Time Markov Decision Processes

2007-11 Klaus Wehrle (editor): 6. Fachgespräch Sensornetzwerke

2007-12 Uwe Naumann: An L-Attributed Grammar for Adjoint Code

2007-13 Uwe Naumann, Michael Maier, Jan Riehme, and Bruce Christianson: Second-Order Adjoints by Source Code Manipulation of Numerical Programs

2007-14	Jean Utke, Uwe Naumann, Mike Fagan, Nathan Tallent, Michelle Strout, Patrick Heimbach, Chris Hill, and Carl Wunsch: OpenAD/F: A Modular, Open-Source Tool for Automatic Differentiation of Fortran Codes
2007-15	Volker Stolz: Temporal assertions for sequential and concurrent programs
2007-16	Sadeq Ali Makram, Mesut Güneç, Martin Wenig, Alexander Zimmermann: Adaptive Channel Assignment to Support QoS and Load Balancing for Wireless Mesh Networks
2007-17	René Thiemann: The DP Framework for Proving Termination of Term Rewriting
2007-18	Uwe Naumann: Call Tree Reversal is NP-Complete
2007-19	Jan Riehme, Andrea Walther, Jörg Stiller, Uwe Naumann: Adjoints for Time-Dependent Optimal Control
2007-20	Joost-Pieter Katoen, Daniel Klink, Martin Leucker, and Verena Wolf: Three-Valued Abstraction for Probabilistic Systems
2007-21	Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre: Compositional Modeling and Minimization of Time-Inhomogeneous Markov Chains
2007-22	Heiner Ackermann, Paul W. Goldberg, Vahab S. Mirrokni, Heiko Röglin, and Berthold Vöcking: Uncoordinated Two-Sided Markets
2008-01 *	Fachgruppe Informatik: Jahresbericht 2007
2008-02	Henrik Bohnenkamp, Marielle Stoelinga: Quantitative Testing
2008-03	Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, Harald Zankl: Maximal Termination
2008-04	Uwe Naumann, Jan Riehme: Sensitivity Analysis in Sisyphe with the AD-Enabled NAGWare Fortran Compiler
2008-05	Frank G. Radmacher: An Automata Theoretic Approach to the Theory of Rational Tree Relations
2008-06	Uwe Naumann, Laurent Hascoet, Chris Hill, Paul Hovland, Jan Riehme, Jean Utke: A Framework for Proving Correctness of Adjoint Message Passing Programs
2008-07	Alexander Nyßen, Horst Lichter: The MeDUSA Reference Manual, Second Edition
2008-08	George B. Mertzios, Stavros D. Nikolopoulos: The λ-cluster Problem on Parameterized Interval Graphs
2008-09	George B. Mertzios, Walter Unger: An optimal algorithm for the k-fixed-endpoint path cover on proper interval graphs
2008-10	George B. Mertzios, Walter Unger: Preemptive Scheduling of Equal-Length Jobs in Polynomial Time
2008-11	George B. Mertzios: Fast Convergence of Routing Games with Splittable Flows
2008-12	Joost-Pieter Katoen, Daniel Klink, Martin Leucker, Verena Wolf: Abstraction for stochastic systems by Erlang's method of stages
2008-13	Beatriz Alarcón, Fabian Emmes, Carsten Fuhs, Jürgen Giesl, Raúl Gutiérrez, Salvador Lucas, Peter Schneider-Kamp, René Thiemann: Improving Context-Sensitive Dependency Pairs
2008-14	Bastian Schlich: Model Checking of Software for Microcontrollers
2008-15	Joachim Kneis, Alexander Langer, Peter Rossmanith: A New Algorithm for Finding Trees with Many Leaves

2008-16 Hendrik vom Lehn, Elias Weingärtner and Klaus Wehrle: Comparing recent network simulators: A performance evaluation study

2008-17 Peter Schneider-Kamp: Static Termination Analysis for Prolog using Term Rewriting and SAT Solving

2008-18 Falk Salewski: Empirical Evaluations of Safety-Critical Embedded Systems

2009-03 Alexander Nyßen: Model-Based Construction of Embedded Real-Time Software - A Methodology for Small Devices

2009-05 George B. Mertzios, Ignasi Sau, Shmuel Zaks: A New Intersection Model and Improved Algorithms for Tolerance Graphs

* These reports are only available as a printed version.

Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.