# A Derandomization Using Min-Wise Independent Permutations

Andrei Z. Broder[1], Moses Charikar[2*], and Michael Mitzenmacher[1]

[1] Compaq SRC, 130 Lytton Avenue, Palo Alto, CA 94301, USA.
{broder,michaelm}@pa.dec.com
[2] Computer Science Dept., Stanford University, CA 94305, USA.
moses@cs.stanford.edu

**Abstract.** Min-wise independence is a recently introduced notion of limited independence, similar in spirit to pairwise independence. The later has proven essential for the derandomization of many algorithms. Here we show that approximate min-wise independence allows similar uses, by presenting a derandomization of the RNC algorithm for approximate set cover due to S. Rajagopalan and V. Vazirani. We also discuss how to derandomize their set multi-cover and multi-set multi-cover algorithms in restricted cases. The multi-cover case leads us to discuss the concept of $k$-*minima-wise* independence, a natural counterpart to $k$-wise independence.

## 1 Introduction

Carter and Wegman [6] introduced the concept of universal hashing in 1979, with the intent to offer an input independent, constant average time algorithm for table look-up. Although hashing was invented in the mid-fifties, when for the first time memory become "cheap" and therefore sparse tables became of interest, up until the seminal paper of Carter and Wegman the premise of the theory and practice of hashing was that either the input is chosen at random or the hash function is chosen uniformly at random among all possible hash functions. Both premises are clearly unrealistic: inputs are not random, and the space needed to store a truly random hash function would dwarf the size of the table. Carter and Wegman showed that, in order to preserve the desirable properties of hashing, it suffices to pick the hash function from what is now called a pair-wise independent family of hash functions. Such families of small size exist, and can be easily constructed.

Since then, pairwise independence and more generally $k$-wise independence have proven to be powerful algorithmic tools with significant theoretical and practical applications. (See the excellent survey by Luby and Wigderson [11]

* Supported by the Pierre and Christine Lamond Fellowship and in part by an ARO MURI Grant DAAH04-96-1-0007 and NSF Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

and references therein.) One important theoretical application of pairwise independence is for the derandomization of algorithms. A well-known example is to find a large cut in a graph. One can color the vertices of a graph with $|E|$ edges randomly using two colors, the colors being determined by a pairwise independent hash function chosen at random from a small family. The colors define a cut, and on average the cut will have $|E|/2$ crossing edges. Hence, by trying every hash function in the family one finds a cut with at least the expected number of crossing edges, $|E|/2$.

Recently, we introduced an alternative notion of limited independence based on what we call min-wise independent permutations [4]. Our motivation was the connection to an approach for determining the resemblance of sets, which can be used for example to identify documents on the World Wide Web that are essentially the same [2, 3, 5]. In this paper we demonstrate that the notion of min-wise independence can also prove useful for derandomization. Specifically, we use a polynomial-sized construction of approximate min-wise independent permutations due to Indyk to derandomize the parallel approximate set cover algorithm of Rajagopalan and Vazirani [12]. (From now on, called the *RV-algorithm.*) This example furthers our hope that min-wise independence may prove a generally useful concept.

The paper proceeds as follows: in Section 2, we provide the definitions for min-wise and approximately min-wise independent families of permutations. We also state (without proof) Indyk's results. In Section 3, we provide the necessary background for the RV-algorithm. In particular, we emphasize how the property of min-wise independence plays an important role in the algorithm. In Section 4, we demonstrate that the RV-algorithm can be derandomized using a polynomial sized approximately min-wise independent family. Finally, in Section 5, we briefly discuss how to extend the derandomization technique to the set multi-cover and multi-set multi-cover algorithms proposed by Rajagopalan and Vazirani. This discussion motivates a generalization of min-wise independence to *$k$-minima-wise* independence, a natural counterpart to $k$-wise independence.

## 2 Min-wise independence

We provide the necessary definitions for min-wise independence, based on [4].

Let $S_n$ be the set of all permutations of $[n]$. We say that $\mathcal{F} \subseteq S_n$ is *exactly min-wise independent* (or just *min-wise independent* where the meaning is clear) if for any set $X \subseteq [n]$ and any $x \in X$, when $\pi$ is chosen at random [1] from $\mathcal{F}$ we have

$$\mathbf{Pr}\big(\min\{\pi(X)\} = \pi(x)\big) = \frac{1}{|X|}. \tag{1}$$

In other words we require that all the elements of any fixed set $X$ have an equal chance to become the minimum element of the image of $X$ under $\pi$.

[1] To simplify exposition we shall assume that $\pi$ is chosen uniformly at random from $\mathcal{F}$, although it could be advatageous to use a another distribution instead. See [4].

We say that $\mathcal{F} \subseteq S_n$ is *approximately min-wise independent with relative error* $\epsilon$ (or just approximately min-wise independent where the meaning is clear) if for any set $X \subseteq [n]$ and any $x \in X$, when $\pi$ is chosen at random from $\mathcal{F}$ we have

$$\left| \mathbf{Pr}\big(\min\{\pi(X)\} = \pi(x)\big) - \frac{1}{|X|} \right| \leq \frac{\epsilon}{|X|}. \tag{2}$$

In other words we require that all the elements of any fixed set $X$ have only an almost equal chance to become the minimum element of the image of $X$ under $\pi$.

Indyk has found a simple construction of approximately min-wise independent permutations with useful properties for derandomization [9]. The construction is derived from a family of hash functions that map $[n]$ to a larger set $[m]$ and have certain limited independence properties. A function $h : [n] \to [m]$ induces a permutation $\pi$ of $[n]$ as follows: sort the $n$ pairs $(h(x), x)$ for $x \in [n]$ in lexicographic order and define $\pi(x)$ to be the index of $(h(x), x)$ in the sorted order. Thus, a family of hash functions $\{h : [n] \to [m]\}$ induces a family of permutations of $[n]$. Indyk's results imply the following proposition.

**Proposition 1. [Indyk]** *There exists constants $c_1$ and $c_2$ such that, for any $c_1 \log(1/\epsilon)$-wise independent family $\mathcal{H}$ of hash functions from $[n]$ to $[c_2 n/\epsilon]$, the family of permutations on $[n]$ induced by $\mathcal{H}$ is approximately min-wise independent with relative error $\epsilon$.*

Using the above proposition, an approximately min-wise independent family can be constructed as follows. Let $r = \lceil \log(c_2 n/\epsilon) \rceil$. We need a hash function that associates to each element in $[n]$ an $r$-bit string. We construct a string of length $n \cdot r$ bits, representing the concatenation of all the hash values, such that the bits are $rc_1 \log(1/\epsilon)$ independent. Thus the hash values for any $c_1 \log(1/\epsilon)$ elements are independent. Proposition 1 ensures that the family of permutations induced by this construction is approximately min-wise independent. Moreover, we can use the constructions of almost $k$-wise independent random variables due to Alon et. al. [1]. The fact that the bits will be only approximately $k$-wise independent can be absorbed into the relative error for the approximately min-wise independent family of permutations. As noted in [1], the construction of the appropriate approximately independent bit strings can be performed in NC, implying that the construction of an approximately min-wise independent family of permutations can be performed in NC. The size of the family of permutations obtained is $n^{O(\log(1/\epsilon))}$.

Hence in what follows we will use the fact that there exist NC-constructible approximately min-wise independent families of permutations of size $n^{O(\log(1/\epsilon))}$.

## 3 The parallel set cover algorithm

### 3.1 The problem

The set cover problem is as follows: given a collection of sets over a universe of $n$ elements, and given an associated cost for each set, find the minimum cost

Preprocess.

**Iteration:**

For each not-covered element $e$, compute $value(e)$.

For each set $S$, include $S$ in $\mathcal{L}$ if $\sum_{e \in U(S)} value(e) \geq C_S/2$.

**Phase:**

(a) Permute $\mathcal{L}$ at random.

(b) Each not-covered element $e$ votes for the first set $S$ in the random order such that $e \in S$.

(c) If $\sum_{e \text{ votes } S} value(e) \geq C_S/16$, then add $S$ to the set cover.

(d) Remove from $\mathcal{L}$ any set not satisfying $\sum_{e \in U(S)} value(e) \geq C_S/2$.

Repeat until $\mathcal{L}$ is empty.

Iterate until all elements are covered.

**Fig. 1.** The RV-algorithm for parallel set cover

sub-collection of sets that covers all of the $n$ elements. This problem (with unit costs) is included in Karp's famous 1972 list [10] of NP-complete problems. (See also [8].)

The natural greedy algorithm repeatedly adds to the cover the set that minimizes the average cost per newly added element. In other words, if the cost of set $S$ is $C_S$, then at each step we add the set that minimizes $C_S/|U(S)|$, where $U(S)$ is the subset of $S$ consisting of elements not yet covered. The greedy algorithm yields an $H_n$ factor approximation. ($H_n$ denotes the harmonic number $\sum_{1 \leq i \leq n} 1/i$.) For more on the history of this problem, see [12] and references therein. In particular Feige [7] has shown that improving this approximation is unlikely to be computationally feasible.

### 3.2 A parallel algorithm

The RV-algorithm is a natural modification of the greedy algorithm: instead of repeatedly choosing the set that covers elements at the minimum average current-cost, repeatedly choose some sets randomly from all sets with a suitably low minimum average current-cost. The intuition is that choosing several sets at a time ensures fast progress towards a solution; randomness is used in an ingenious way to ensure a certain amount of coordination so that not too many superfluous sets (that is, sets that cover few, if any, new elements) are used.

Define the *value* of an element to be:

$$value(e) = \min_{S \ni e} \frac{C_S}{|U(S)|}.$$

That is, the value of an element is the minimum possible cost to add it to the current cover. The algorithm of Rajagopalan and Vazirani is depicted in Figure 3.1.

The preprocessing step is used to guarantee that the costs $C_S$ lie in a limited range; this is not of concern here since it does not involve any randomization. The randomization comes into play when the sets of $\mathcal{L}$ are randomly permuted, and each element votes for the first set in the random order. This property is exploited in the analysis of the algorithm in two ways:

1. The set that each element votes for is equally likely to be any set that contains it.
2. Given any pair of elements $e$ and $f$, let $N_e$ be the number of sets containing $e$ but not $f$, let $N_f$ be the number of sets containing $f$ but not $e$, and let $N_b$ be the number of sets that contain both. The probability that both $e$ and $f$ vote for the same set is
$$\frac{N_b}{N_e + N_b + N_f}.$$

Interestingly, both of these properties would hold if $\mathcal{L}$ were permuted according to a min-wise independent family of permutations; in fact, this is all that is required in the original analysis. Hence if we had a polynomial sized min-wise independent family, we could derandomize the algorithm immediately. Unfortunately, the lower bounds proven in [4] show that no such family exists; any min-wise independent family would have size exponential in $|\mathcal{L}|$.

We therefore consider what happens when we replace step (a) of the parallel set cover algorithm with the following step:

(a') Permute $\mathcal{L}$ using a random permutation from an approximately min-wise independent family with error $\epsilon$.

As we shall explain, for suitably small $\epsilon$ this replacement does not affect the correctness of the algorithm, and the running time increases at most by a constant factor. Using this fact, we will be able to derandomize the algorithm using Indyk's polynomial-sized construction.

## 4 The derandomization

We note that the proof of the approximation factor of the algorithm, as well as the bound on the number of iterations, does not change when we change how the permutation on $\mathcal{L}$ is chosen. Hence we refer the interested reader to the proofs in [12], and consider only the crux of the argument for the derandomization, namely the number of phases necessary for each iteration.

As in [12], we establish an appropriate potential function $\Phi$, and show that its expected decrease $\Delta\Phi$ in each phase is $c\Phi$ for some constant $c$. The potential function is such that if it ever becomes 0 we are done. In [12], this was used to show that $O(\log n)$ phases per round are sufficient, with high probability. By using a polynomial sized family of approximately min-wise independent permutations, we can try all possible permutations (on a sufficiently large number of processors) in each phase; in this way we ensure that in each phase the potential $\Phi$ decreases by a constant factor. This derandomizes the algorithm.

We review the argument with the necessary changes. The potential function $\Phi$ is $\sum_S U(S)$. The degree of an element $e$, denoted $\deg(e)$ is the number of sets containing it. A set-element pair $(S, e)$ with $e \in U(S)$ is called *good* if $\deg(e) \geq \deg(f)$ for at least $3/4$ of the elements $f \in U(S)$. We show that on average a constant fraction of the good $(S, e)$ pairs disappear in each phase (because sets are added to the cover), from which we can easily show that $\mathbf{E}(\Delta\Phi) \geq c\Phi$.

**Lemma 2.** *Let $e, f \in U(S)$ with $\deg(e) \geq \deg(f)$. Then*

$$\mathbf{Pr}(f \textit{ votes for } S \mid e \textit{ votes for } S) > \frac{1-\epsilon}{2(1+\epsilon)}.$$

*Proof.* Let $N_e$ be the number of sets containing $e$ but not $f$, let $N_f$ be the number of sets containing $f$ but not $e$, and let $N_b$ be the number of sets that contain both. The set $S$ is chosen by both $e$ and $f$ if it the smallest choice for both of them; this happens with probability at least $\frac{1-\epsilon}{N_e+N_b+N_f}$, by the definition of approximate min-wise independence. Similarly, the set $S$ is chosen by $e$ with probability at most $\frac{1+\epsilon}{N_e+N_b}$. Hence

$$\mathbf{Pr}(f \text{ votes for } S \mid e \text{ votes for } S) \geq \frac{1-\epsilon}{1+\epsilon} \cdot \frac{N_e+N_b}{N_e+N_b+N_f} \geq \frac{1-\epsilon}{2(1+\epsilon)}.$$

The last inequality follows from the fact that $N_e \geq N_f$.

The above lemma suggests that if $(S, e)$ is good, and $e$ votes for $S$, then $S$ should get many votes. Indeed, this is the case.

**Lemma 3.** *If $(S, e)$ is good then*

$$\mathbf{Pr}(S \textit{ is picked } \mid e \textit{ votes for } S) > \frac{1-4\epsilon}{15}.$$

*Proof.* Clearly $\textit{value}(f) \leq C_S/|U(S)|$ for any $f \in U(S)$, so

$$\sum_{\substack{f \in U(S) \\ \deg(f) > \deg(e)}} \textit{value}(f) \leq \frac{C_S}{4} .$$

But if $S \in \mathcal{L}$, then $\sum_{f \in U(S)} \textit{value}(f) \geq C_S/2$. Therefore

$$\sum_{\substack{f \in U(S) \\ \deg(f) \leq \deg(e)}} \textit{value}(f) \geq \frac{C_S}{4} .$$

By Lemma 2, if $e$ votes for $S$, then each $f$ with $\deg(f) \leq \deg(e)$ votes for $S$ with probability at least $(1-\epsilon)/(2(1+\epsilon))$. Hence, conditioned on $e$ voting for $S$, the expected total value of all elements that vote for $S$ is at least $C_S(1-\epsilon)/\big(8(1+\epsilon)\big)$. Let $p$ be the probability that $S$ is picked in this case. Then as the total value from all elements that vote for $S$ is at most $C_S$, clearly

$$pC_S + (1-p)\frac{C_S}{16} \geq \frac{C_S(1-\epsilon)}{8(1+\epsilon)}.$$

From this we obtain that $p > (1-4\epsilon)/15$.

From the Lemma above we show that the expected decrease in the potential function is a constant fraction per round.

**Lemma 4.** $\mathbf{E}(\Delta\Phi) \geq \frac{(1-5\epsilon)}{60}\Phi$.

*Proof.* As in [12], we estimate the decrease in $\Phi$ due to each pair $(S, e)$ when $e$ votes for $S$ and $S$ joins the cover. The associated decrease is $\deg(e)$ since $\Phi$ decreases by one for every remaining set that contains $e$. Hence

$$\begin{aligned}
\mathbf{E}(\Delta\Phi) &\geq \sum_{(S,e):e\in U(S)} \mathbf{Pr}(e \text{ voted } S \text{ and } S \text{ was picked }) \cdot \deg(e) \\
&\geq \sum_{(S,e) \text{ good}} \mathbf{Pr}(e \text{ voted } S) \cdot \mathbf{Pr}(S \text{ was picked} \mid e \text{ voted } S) \cdot \deg(e) \\
&\geq \sum_{(S,e) \text{ good}} \frac{1-\epsilon}{\deg(e)} \cdot \frac{1-4\epsilon}{15} \cdot \deg(e) \geq \sum_{(S,e) \text{ good}} \frac{1-5\epsilon}{15} \\
&\geq \sum_{(S,e):e\in U(S)} \frac{1-5\epsilon}{60} \geq \frac{1-5\epsilon}{60}\Phi.
\end{aligned}$$

If initially we have $n$ sets and $m$ elements, then initially $\Phi \leq mn$, and hence we may conclude that at most $O(\log nm)$ phases are required before an iteration completes. Given the results of [12], we may conclude:

**Theorem 5.** *The algorithm* Parallel Set Cover *can be derandomized to an* $NC^3$ *algorithm that approximates set cover within a factor of* $16H_n$ *using a polynomial number of processors.*

One may trade of the number of processors and a constant factor in the running time by varying the error $\epsilon$. However, the family must be sufficiently large so that $\epsilon$ is small enough for the analysis to go through. Having $\epsilon < 1/5$ is sufficient (this can be improved easily, at least to $\epsilon < 1/3$).

## 5 Extensions

Besides the parallel set cover algorithm, Rajagopalan and Vazirani also provide algorithms for the more general set multi-cover and multi-set multi-cover problems. In the set multi-cover problem, each element has a requirement $r_e$, and it must be covered $r_e$ times. In the multi-set multi-cover problem, multi-sets are allowed. These algorithms follow the same basic paradigm as the parallel set cover algorithm, except that during the algorithm an element that still needs to be covered $r(e)$ more times gets $r(e)$ votes. (Note $r(e)$ is dynamic; $r(e) = r_e$ initially.)

Our derandomization approach using approximately min-wise independent families of permutations generalizes to these extensions as well, subject to a technical limitation that the initial requirements $r_e$ must be bounded by a fixed constant. We need slightly more than approximate min-wise independence, however. The following properties are sufficient[2]:

- the ordered $r(e)$-tuple of the first $r(e)$ sets containing an element $e$ in the random order is equally likely to be any ordered $r(e)$-tuple of sets that contain $e$,
- for any pair of elements $e$ and $f$ both in some set $S$, the ordered $(r(e) + r(f) - 1)$-tuple of the first $r(e) + r(f) - 1$ sets containing either $e$ or $f$ in the random order is equally likely to be any ordered $(r(e) + r(f) - 1)$-tuple of sets that contain either $e$ or $f$.

Note that when $r(e) = r(f) = 1$, these conditions are implied by min-wise independence, as we would expect.

These requirements suggest a natural interpretation of min-wise independence: suppose that not just any element of a set $X$ was equally likely to be the first after applying a permutation, but that any ordered set of $k$ elements of a set $X$ are equally likely to be the first $k$ elements (in the correct order) after applying a permutation to $X$. Let us call this $k$-minima-wise independence. Then the properties above correspond to $\max_{e,f}(r(e) + r(f) - 1)$-minima-wise independence; if $\max_e r(e)$ is a fixed constant, then we require a $k$-minima-wise independent family of permutations for some constant $k$. In fact, as with the parallel set cover problem, we require only approximate $k$-minima-wise independence, and the construction of Indyk can be generalized to give us an appropriate family of polynomial size when $k$ is a constant.

We note in passing that for estimating the resemblance of documents as in [2] and [5] with a "sketch" of size $k$ we need one sample from a $k$-minima-wise independent family, while for the method presented in [3], we need $k$ separate samples from a min-wise independent family.

There is an interesting meta-principle behind our derandomizations, which appears worth emphasizing here.

*Remark 6.* Let $\mathcal{E}$ be an event that depends only on the order of the first $k$ elements of a random permutation. Then any bound on the probability of $\mathcal{E}$ that holds for random permutations also holds for any $k$-minima-wise independent family. Moreover, for any approximately $k$-minima-wise independent family, a suitable small correction to the bound holds.

[2] In fact they are more than is necessary; however, stating the properties in this form is convenient.

For example, many of the lemmata in [12] prove bounds for events assuming that the random permutations are generated by assigning each set a uniform random variable from $[0,1]$ and then sorting. Because the events these lemmata bound depend only on the first $(r(e)+r(f)-1)$ sets of the permutation, the lemmata still hold when using $(r(e)+r(f)-1)$-minima-wise independent families, and only minor corrective terms need to be introduced for $(r(e)+r(f)-1)$-minima-wise independent families. Hence given the results of [12], the derandomizations follow with relatively little work.

## 6 Conclusion

We have demonstrated a novel derandomization using the explicit construction of approximate min-wise independent families of permutations of polynomial size. We expect that this technique may prove useful for further derandomizations.

The question of how to best construct small approximately min-wise independent families of permutations remains open. Improvements in these constructions would lead to improvements in the number of processors required for our derandomizations here, and more generally may enhance the utility of this technique.

## References


1. N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost $k$-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–304, 1992.
2. A. Z. Broder. On the resemblance and containment of documents. In *Proceedings of Compression and Complexity of Sequences 1997*, pages 21–29. IEEE Computer Society, 1988.
3. A. Z. Broder. Filtering near-duplicate documents. In *Proceedings of FUN 98*, 1998. To appear.
4. A. Z. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, pages 327–336, 1998.
5. A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the Web. In *Proceedings of the Sixth International World Wide Web Conference*, pages 391–404, 1997.
6. J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, Apr. 1979.
7. U. Feige. A threshold of $\ln n$ for approximating set cover (preliminary version). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 314–318, Philadelphia, Pennsylvania, 22–24 May 1996.
8. M. R. Garey and D. S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.
9. P. Indyk, A small approximately min-wise independent family of hash functions. manuscript, 1998.
10. R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–104. Plenum Press, New York, 1972.

11. M. Luby and A. Wigderson. Pairwise independence and derandomization. Technical Report TR-95-035, International Computer Science Institute, Berkeley, California, 1995.
12. S. Rajagopalan and V. V. Vazirani. Primal-dual RNC approximation algorithms for (multi)-set (multi)-cover and covering integer programs. In *34th Annual Symposium on Foundations of Computer Science*, pages 322–331, Palo Alto, California, 3–5 Nov. 1993. IEEE. Journal version to appear in *SIAM Journal of Computing*.