1 Derandomization

K. Gopalakrishnan and D. R. Stinson

1.1 Monte Carlo Algorithms

A *decision problem* is a problem which has a "yes" or "no" answer for every problem instance I.

A *yes-biased Monte Carlo Algorithm* having error probability ϵ is a randomized algorithm to solve a decision problem such that the following conditions are satisfied:

1. If the instance I is a no-instance, then the algorithm answers "no."
2. If the instance I is a yes-instance, then the probability that the algorithm answers "yes" is at least $1 - \epsilon$.

A *no-biased* Monte Carlo Algorithm is defined in the obvious way.

1.1 Example Solovay-Strassen Algorithm for "Composites"

	Input: an odd integer n
1.	choose a random integer a, $1 \leq a \leq n-1$
2.	**if** $\left(\frac{a}{n}\right) = a^{(n-1)/2} \bmod n$ **then**
3.	answer "no: n is prime"
	else
4.	answer "yes: n is composite"

In the above algorithm, $\left(\frac{a}{n}\right)$ is a Jacobi symbol. The Solovay-Strassen algorithm for "Composites" is a yes-biased Monte Carlo algorithm having error probability $\epsilon \leq 1/2$.

Despite the fact that Monte Carlo algorithms can give wrong answers, the error probability can be made as small as desired by repeated application of the algorithm. Suppose we have a yes-biased algorithm, and we apply the algorithm k times. If we get at least one "yes" answer we can quit, since the instance I must be a yes-instance. On the other hand, if I is a yes-instance, then the probability of getting k "no" answers is at most ϵ^k, which can be made arbitrarily small by taking k large enough.

1.2 Two-Point Based Sampling

The above analysis is based on the assumption that the sequence of random numbers used in the successive trials are independent. In the implementation of randomized algorithms in actual practice, one usually uses a pseudo-random number generator, which invalidates this assumption.

An alternative technique known as *two-point sampling*, recently developed by Chor and Goldreich [3], is amenable to more realistic implementation and analysis. This technique essentially uses orthogonal arrays. Suppose we need $\log n$ random bits for one trial of the algorithm and we want to conduct k trials of the algorithm. Instead of generating $k \log n$ random bits, which will ensure independence, we can use orthogonal arrays to generate pairwise independent random numbers as follows. Consider any $OA(2, k, n)$. This array has n^2 rows. Generate $2 \log n$ random bits and use them to index a specific row of the OA. Now apply the Monte Carlo algorithm using the k points which are elements of the row selected. It can be proved by elementary combinatorial arguments [4] that the probability of getting k "no" answers for a yes-instance is bounded above by

$$\frac{\epsilon}{1+(k-1)(1-\epsilon)}.$$

1.2 Example An application of the two-point sampling technique, to the problem of random pattern VLSI chip testing, deserves special mention due to its practical importance. Consider a combinational circuit with n inputs. One can test the circuit by applying all 2^n possible test vectors to the circuit and checking for the response of the circuit. As the number of test patterns is too high for exhaustive testing, usually the circuit is tested with some smaller number of random input patterns. The probability of not detecting a specific fault in the circuit can then be analyzed. But the analysis is invalid if successive test vectors applied are not independent.

In view of this problem, T. Spenser [7] developed a pseudo-random pattern generator based on the idea of two-point sampling. This generator can be efficiently implemented as a modification of the classical LFSR, and it is amenable to a realistic analysis. This generator is, in fact, an orthogonal array $OA(2, t, 2^n)$.

1.3 Total Derandomization

In some randomized algorithms for search problems, it is possible to completely eliminate the random bits needed, thus converting the algorithm into a deterministic algorithm. This process is called (total) derandomization. Using orthogonal arrays of strength t, derandomization can be accomplished for randomized algorithms whose analysis only depends on t-wise rather than fully independent random choices. This method was first developed by M. Luby [6] for the case $t = 2$ in the context of the maximal independent set problem and later generalized to arbitrary constant t by Alon, Babai and Itai [1].

1.3 Example We illustrate the idea using the "Monochromatic K_4's" problem found in [2].

For any integer $n \geq 4$, there is an edge-coloring of K_n with two colors such that the number of monochromatic induced K_4's is at most $\binom{n}{4}/32$. This can be seen easily using a probabilistic argument as follows. For a 2-edge-coloring c of K_n, let $M(c)$ denote the number of monochromatic induced K_4's. Let $\overline{M}$ denote the average value of $M(c)$ over all colorings c. We have

$$\begin{aligned}
\overline{M} \times 2^{\binom{n}{2}} &= \sum_c M(c) \\
&= \sum_{A \subseteq V, |A|=4} |\{c : |c(A)| = 1\}| \\
&= \binom{n}{4} \times 2 \times 2^{\binom{n}{2}-6} \\
&= \binom{n}{4} \times 2^{\binom{n}{2}-5}
\end{aligned}$$

Hence $\overline{M} = \binom{n}{4}/32$. So there must exist a coloring with at most $\binom{n}{4}/32$ monochromatic induced K_4's.

This proof is existential rather than constructive. How would one actually find such a coloring? A randomized algorithm would simply generate a random coloring and check whether it has the desired property. Since the total number of possible colorings, $2^{\binom{n}{2}}$, is exponential in n, it is not possible to test every coloring in polynomial time. Hence, there is no guarantee that a suitable coloring will be found in time $poly(n)$.

However, it is not necessary that the color of every edge be chosen independently at random. It suffices to consider a sample space $\mathbf{S}$ where every subset of six edges has colors chosen independently at random. Since we need only a 6-wise independent sample space, we can use the rows of an $\mathrm{OA}_\lambda\left(6, \binom{n}{2}, 2\right)$ as our sample space $\mathbf{S}$. A deterministic algorithm will check the colorings determined by all the rows of the OA, until one is found in which the number of monochromatic induced K_4's is at most $\binom{n}{4}/32$.

The proof that this approach will work is as follows: Within the sample space $\mathbf{S}$, we once again have

$$\begin{aligned}\overline{M} \times \lambda 2^6 &= \sum_{c \in \mathbf{S}} M(c) \\ &= \sum_{A \subseteq V, |A|=4} |\{c \in \mathbf{S} : |c(A)| = 1\}| \\ &= \binom{n}{4} \times 2\lambda\end{aligned}$$

and so $\overline{M} = \binom{n}{4}/32$, as before.

Now, it is possible to deterministically check any particular coloring in polynomial time. Thus, we will have a polynomial-time deterministic algorithm provided that $\lambda = poly(n)$. This can be accomplished as follows. Let m be such that $\binom{n}{2} \leq 2^m - 1$. Then we can construct an orthogonal array $\mathrm{OA}_{2^{3(m-2)}}\left(6, \binom{n}{2}, 2\right)$ as the dual code of a BCH code with designed distance 7. The size of the sample space is $O(n^6)$ and so it is a $poly(n)$-sized sample space.

More generally, suppose that the analysis of a randomized algorithm does not require the complete independence of n random bits, but only t-wise independence. Then, one can use any $OA_\lambda(t, n, 2)$ as the sample space instead of the sample space with 2^n points. As in the above example, appropriate sample spaces can be obatined from BCH codes with designed distance $t + 1$, as follows.

1.4 **Theorem** Suppose $n = 2^m - 1$ and $t = 2e$. Then there exists an orthogonal array $OA_{2^{e(m-2)}}(2e, n, 2)$, which yields a t-wise independent sample space of size $(n+1)^e$.

This approach can also be parallelized in an obvious way. Thus, it yields both a $poly(n)$-time sequential algorithm, and a $polylog(n)$-time parallel algorithm with $poly(n)$ processors.

1.4 See Also

§??	for information on orthogonal arrays of strength 2
§??	for information on orthogonal arrays of strength $t > 2$
§??	for information on BCH codes
[5]	for a general survey of randomized algorithms
[8]	for a collection of references on pairwise independence and derandomization techniques

References

[1] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, **7** (1986), 567–583.

[2] N. Alon and J. H. Spencer. *The Probabilistic Method.* John Wiley & Sons, Inc., 1992.

[3] B. Chor and O. Goldreich. On the power of two-point based sampling. *Journal of Complexity*, **5** (1989), 96–106.

[4] K. Gopalakrishnan and D. R. Stinson. A simple analysis of the error probability of two-point based sampling. Preprint.

[5] R. Gupta, S. A. Smolka, and S. Bhaskar. On randomization in sequential and distributed algorithms. *ACM Computing Surveys*, **26** (1994), 7–86.

[6] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, **15** (1986), 1036–1053.

[7] T. H. Spenser. Provably good pattern generators for a random pattern test. *Algorithmica*, **11** (1994), 429–442.

[8] A. Wigderson. The amazing power of pairwise independence. *Proceedings of the 26th ACM Symposium on the Theory of Computing*, 1994, pp. 645–647.