A Randomized Algorithm for the Union of Sets Problem*

ZHANG Li-yu, ZHU Hong, ZHANG Pi-xing

(Department of Computer Science, Fudan University, Shanghai 200433, China)

E-mail: hzhu@fudan.edu.cn

http://www.fudan.edu.cn

Received April 15, 1999; accepted September 21, 1999

Abstract: Randomized algorithms are playing a more and more important role in computation because of their simplicity and fastness. But sometimes the good performance of randomized algorithms does not require completely independent random variables as their input. In this paper, a new random algorithm is introduced for the classical problem of estimating the cardinality of a union of sets, which only needs pair-wise independent random input. This approach helps to reduce the random bits used in the algorithm. For fixed accuracy parameter ε and confidence parameter δ, the algorithm needs $O(t^{1/2})$ random bits, much fewer than those of a standard randomized algorithm $O(t\log tM)$. And the running time bounds of the algorithm do not increase essentially ($O(t^2\log M)$, where t is the number of sets and M is the maximal cardinality of an individual set).

Key words: randomized algorithm, pair-wise independence, union of sets, random bit, k-wise independence.

In recent years, the role of randomness in computation has become more and more dominant. Randomized algorithms are often preferred for their somplicity and fastness. Generally, we can view a randomized algorithm as a deterministic algorithm with two inputs: "true input" and "random input". As the "true input" is just the same as any deterministic algorithm, randomized algorithms need "real" random inputs to achieve good performance. Also for this sake, it is commonly suggested that different executions of a randomized algorithm have mutually independent random inputs. However, sometimes randomized algorithms "behave as well" if only "certain" rather than "full" independence is satisfied. In such situations, we can generate random inputs from a much smaller sample space which yields a saving in the number of the random bits used (we suppose random inputs are represented in binary form). This is important because "real" random bits are hard (if not impossible) to generate in actual machines. So the number of random bits used in an algorithm should be regarded as a kind of cost like time and space which we expect to use as less as possible. In fact, there is a trade-off between randomness and time in a randomized algorithm just as between time and space. The problem was first raised by Karp and Pippenger[1].

Reference [2] demonstrated a general method for constructing small sample space that satisfies certain

* This project is supported by the National Natural Science Foundation of China under Grant No. 69673038 (国家自然科学基金). **ZHANG Li-yu** was born in 1974. He is a graduate student of Department of Computer Science, Fudan University. His current research areas include the design and analysis of algorithms, computational complexity. **ZHU Hong** was born in 1939. He is a professor and doctoral supervisor of Department of Computer Science, Fudan University. His current research areas include the design and analysis of algorithms, computational complexity, graph drawing, cryptology, data visualization. **ZHANG Pi-xing** was born in 1975. He is a graduate student of Department of Computer Science, Fudan University. His current research areas include the design and analysis of algorithms, computational complexity.

rather than complete independence. And especially for a kind of so-called k-wise independence sample space, we have explicit constructions in Refs. [3,4]. All these constructions play an important role in derandomization.

A series of random variables is said to be k-wise independent if any k of them are mutually independent. An important special case is pair-wise ($k=2$) independence, which has found many applications in computer science, Ref. [5] is a comprehensive survey on this approach. Among them, Luby[6] showed a good example that took advantage of pair-wise independence in converting parallel Monte Carlo algorithms into deterministic ones.

Reference [7] presents another simple approach to generate pair-wise independent variables. In this paper, we generalize the approach a bit to extract long pair-wise independent random variables from a few completely independent ones. Applying this to the Union of Sets problem, we get a random algorithm that has as good performance but uses much fewer random bits than those in Ref. [8].

1 A Simple Random Algorithm for the US Problem

In this section, we first present the **US** problem. Then we give a simple random algorithm for **US** problem due to Ref. [8].

The **US** problem is to find the cardinality of a union of sets, which is a classical combinatorial enumeration problem. There are a number of concrete problems which can be expressed as computing the cardinality[8]. The **US** problem's classical solution is given by the principle of inclusion and exclusion:

$$|S_1 \cup S_2 \cup \ldots \cup S_t| = \sum_i |S_i| - \sum_{i<j} |S_i \cap S_j| + \sum_{i<j<k} |S_i \cap S_j \cap S_k| - \ldots,$$

where $S_j (j=1,2,\ldots,t)$ are a series of sets.

When t is very large the direct evaluation of the inclusion-exclusion sum is not practical since the number of terms is 2^t-1. Now let us consider a randomized algorithm which produces an estimate of the cardinality of a union of sets. This method requires three assumptions:

(1) It is "easy" to determine the cardinality of each set S_j(which we suppose will be done in $O(1)$ time);

(2) An element can be drawn at random from the uniform distribution over any one of the sets (which we suppose will be done in $O(1)$ time);

(3) It is "easy" to test whether a given element lies in a given set S (which we suppose will be done in $O(\log|S|)$ time).

These conditions are often fulfilled in concrete examples of the **US** problem, like in DNF problem[9].

Algorithm 1.[8]

(1) Randomly choose a set S_i with probability $|S_i| \Big/ \sum_{j=1}^{t} |S_j|$;

(2) Uniformly choose an $x \in S_i$;

(3) By testing the membership of x in S_j, determine $\mathrm{cov}(x) = |\{i \mid x \in S_i\}|$;

(4) $X \leftarrow \sum_{j=1}^{t} |S_j| / \mathrm{cov}(x)$.

In this algorithm, we can compute $\mathrm{cov}(x)$ by testing x's membership in each S_j.

And what does the output of this algorithm tell us? We have the following theorem:

Theorem 1. Let X be the random variable obtained by the above algorithm. $E[X]$ and $\sigma[X]$ represent the expection and variance of X respectively. Then we have $E[X]=|\cup S_i|$ and $\sigma[X]^2 \leqslant (t-1)|\cup S_i|^2$.

Proof. Suppose $\cup S_i$ has m distinct elements $x_1, x_2, \ldots, x_m$. Then

$$E[X] = \sum_{i=1}^{m} \Pr[x_i \text{ is chosen}] \Big(\sum_{j=1}^{t} |S_j| / \mathrm{cov}(x_i) \Big).$$

Since $\Pr[x_i \text{ is chosen}]$ is exactly

$$\sum_{l=1}^{\mathrm{cov}(x_i)} \left(|S_{i_l}| \Big/ \sum_{j=1}^{t} |S_j|\right) \cdot 1/|S_{i_l}| = \mathrm{cov}(x_i) \Big/ \sum_{j=1}^{t} |s_j|,$$

where $S_{i_l}(l=1,2,\ldots,\mathrm{cov}(x_i))$ are the sets which contain x_i, we obtain $E[X]=m=|\bigcup S_i|$.

Now we consider $\sigma[X]$. By the property of variance,

$$\begin{aligned}\sigma[X]^2 &= E[X^2]-E[X]^2 = \sum_{i=1}^{m}\Big\{\sum_{j=1}^{t}|S_i|/\mathrm{cov}(x_i)\Big\}^2 \Pr[x \text{ is chosen}] - |\bigcup S_i|^2 \\ &= \sum_{i=1}^{m}\Big(\sum_{j=1}^{t}|S_i|/\mathrm{cov}(x_i)\Big) - |\bigcup S_i|^2 = \Big(\sum_{j=1}^{t}|S_i|\Big)\sum_{i=1}^{m} 1/\mathrm{cov}(x_i) - |\bigcup S_i|^2 \\ &\leqslant (t-1)|\bigcup S_i|^2.\end{aligned}$$

□

We can see now X is an unbiased estimator of the cardinality of the union of sets. Then we might estimate $|\bigcup S_i|$ by taking the average of N samples of the estimator X: $Y=(X_1+X_2+\ldots+X_N)/N$. But how big would N be to guarantee Y deviates much from $|\bigcup S_i|$ with "small" probability? To formalize this, we introduce the following definition:

A randomized algorithm A for combinatorial enumeration problems is called an ϵ,δ-approximation algorithm if for every instance I,

$$\Pr\left[\frac{|C(I)-A(I)|}{C(I)} > \epsilon\right] < \delta, \tag{1}$$

where $C(I)$ is the correct solution and $A(I)$ represents the solution produced by Algorithm A. Often, we refer to such ϵ as accuracy parameter and δ as confidence parameter.

With a routine calculation we can determine the least N to satisfy inequality (1).

Theorem 2. [8] Let X_i, $i=1,2,\ldots,N$, be the outputs of the N independent executions of Algorithm 1 and $Y=(X_1+X_2+\ldots+X_N)/N$, then $N=\lceil 4t\in(2/\delta)/\epsilon^2\rceil$ is sufficient to guarantee

$$\Pr[(|Y|-|\bigcup S_i|)/|\bigcup S_i| > N\epsilon] < \delta. \tag{2}$$

To prove this, we need the following lemma, which is a generalization of the Chernoff bound. The proof is similar to that of the Chernoff bound and the reader may refer to Ref. [9](p. 98).

Lemma 1. Let $X_1,X_2,\ldots,X_N$ be independent random variables over interal $[0,1]$, such that, for $1\leqslant i\leqslant N$, $E[X_i]=p_i$, where $0<p_i<1$. Then for $X=\sum_{i=1}^{N}X_i$, $\mu=E[X]=\sum_{i=1}^{N}p_i$, and any $\delta>0$,

$$\Pr[X>(1+\delta)\mu] < F^+(\mu,\delta),$$

and

$$\Pr[X<(1-\delta)\mu] < F^-(\mu,\delta),$$

where $F^+(\mu,\delta)=\left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\mu}$, $F^-(\mu,\delta)=\exp\left(\frac{-\mu\delta^2}{2}\right)$.

We note that for $0<\delta\leqslant 2e-1$, $F^+(\mu,\delta)<\exp\left(\frac{-\mu\delta^2}{4}\right)$ (Ref. [9], p. 72).

Proof (of Theorem 2). We now apply the above lemma to our problem. Let $X'_i=X_i/T$, where $T=\sum_{j=1}^{t}|S_j|$. It is obvious that X'_is are independent random variables over $[0,1]$ and for all $1\leqslant i\leqslant N$, $E[X_i]=|\bigcup S_i|/T\in[1/t,1]$.

Without loss of generality, suppose $\epsilon<1$, then by Lemma 1

$$\begin{aligned}\Pr[|Y-|\bigcup S_i||>\epsilon|\bigcup S_i|] &= \Pr\left[\left|\sum_{i=1}^{N}X'_i - N|\bigcup S_i|/T\right| > \epsilon\cdot(N|\bigcup S_i|/T)\right] \\ &\leqslant F^+(N|\bigcup S_i|/T,\epsilon)+F^-(N|\bigcup S_i|/T,\epsilon) < 2\exp\left(\frac{-N\epsilon^2}{4}\cdot\frac{|\bigcup S_i|}{T}\right) \\ &\leqslant 2\exp\left(\frac{-N\epsilon^2}{4t}\right).\end{aligned}$$

Since $N=\lceil 4t\ln(2/\delta)/\epsilon^2\rceil$ is sufficient to guarantee $2\exp[-N\epsilon^2/4t]<\delta$, the proof is complete. □

From here on, we will use Algorithm 1' to denote the algorithm which just runs Algorithm 1 $N=\lceil 4t\ln(2/\delta)/\epsilon^2\rceil$ times and takes the mean of the N outputs as its output. Clearly, Algorithm 1' is an ϵ,δ-approximation algorithm for the "union of sets" problem.

2 Pair_Wise Independence

In the above algorithm, it is assumed that all X_is are completely independent. But what will happen if X_is are only pair_wise independent?

Theorem 3. Let X_i and Y be the random variables described as above except that X_i are only pair_wise independent, then $N>(t-1)/\epsilon\delta$ suffices to satisfy (2).

To prove this, we need the following proposition[9]:

Proposition 4. Suppose X_i, $i=1,2,\ldots,m$, be pair_wise independent, then $\sigma\left[\sum_{i=1}^{m}mX_i^2\right]=\sum_{i=1}^{m}\sigma[X_i]^2$.

Proof (of Theorem 3). By the property of expectation and variance, we have

$$E[Y]=E[X_i]=|\cup S_i| \text{ and } \sigma[Y]^2=\sigma[X_i]^2/N\leqslant(t-1)|\cup S_i|^2/N \text{ (by Theorem 1).}$$

Then $\Pr[|Y-|\cup S_i||>\epsilon|\cup S_i|]\leqslant 1/(\epsilon|\cup S_i|/\sigma[Y])^2\leqslant(t-1)/N\epsilon<\delta$, which is a natural result of the well-known Chebyshev inequality. □

One advantage of using pair_wise independent samples is that we need possibly small sample size to achieve certain approximation performance with high probability.

Then to generate N pair_wise independent samples, how many fully independent samples must we have?

Theorem 5. Let $x_i,i=1,2,\ldots,r$, be independent random variables uniformly distributed over Z_p, where p is a prime. Then $y_{i,j,k}=x_i+x_j\cdot k$, $i,j=1,2,\ldots,r$, $k=0,1,\ldots,p-1$, are pair_wise independent variables uniformly distributed over Z_p, where the operations $+$, $\cdot$ are over Z_p.

Proof. First, for any $y_{i,j,k}$ and $a\in Z_p$, $\Pr[y_{i,j,k}=a]=\Pr[x_i+x_j\cdot k=a]=\sum_{b\in Z_p}\Pr[x_j=b]\cdot\Pr[x_i+x_j\cdot k=a|x_j=b]=\sum_{b\in Z_p}\Pr[x_j=b]\cdot\Pr[x_i=a-kb]=1/p$. So, all $y_{i,j,k}$ are uniformly distributed over Z_p.

Now consider two random variables y_{i_1,j_1,k_1} and y_{i_2,j_2,k_2}. They are pair_wise indepenent iff for any $a,b\in Z_p$, $\Pr[y_{i_1,j_1,k_1}=a\wedge y_{i_2,j_2,k_2}=b]=\Pr[y_{i_1,j_1,k_1}=a]\cdot\Pr[y_{i_2,j_2,k_2}=b]=1/p^2$. We prove this in four cases:

Case 1. $i_1=i_2$, $j_1=j_2$ and $k_1\neq k_2$. Then

$$\Pr[y_{i_1,j_1,k_1}=a\wedge y_{i_2,j_2,k_2}=b]=\Pr[x_{i_1}+x_{j_1}\cdot k_1=a\wedge x_{i_2}+x_{j_2}\cdot k_2=b]=\Pr[x_{i_1}=c\wedge x_{j_1}=d]=1/p^2,$$

where $\begin{pmatrix}c\\d\end{pmatrix}$ is the unique solution of the linear equation $\begin{pmatrix}1&k_1\\1&k_2\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}=\begin{pmatrix}a\\b\end{pmatrix}$ over Z_p.

Case 2. $i_1=i_2$, $j_1\neq j_2$. Then

$$\Pr[y_{i_1,j_1,k_1}=a\wedge y_{i_2,j_2,k_2}=b]=\Pr[x_{i_1}+x_{j_1}\cdot k_1=a\wedge x_{i_2}+x_{j_2}\cdot k_2=b]$$

$$=\sum_{c\in Z_p}\Pr[x_{i_1}=c]\cdot\Pr[x_{j_1}=d_1\wedge x_{j_2}=d_2]=\sum_{c\in Z_p}(1/p)(1/p^2)=1/p^2,$$

where d_1 and d_2 are the unique solutions of the two equations $c+x\cdot k_1=a$ and $c+x\cdot k_2=b$ respectively.

Case 3. $i_1\neq i_2$, $j_1=j_2$. Then

$$\Pr[y_{i_1,j_1,k_1}=a\wedge y_{i_2,j_2,k_2}=b]=\Pr[x_{i_1}+x_{j_1}\cdot k_1=a\wedge x_{i_2}+x_{j_2}\cdot k_2=b]$$

$$=\sum_{d\in Z_p}\Pr[x_{j_1}=d]\cdot\Pr[x_{i_1}=a-dk_1\wedge x_{i_2}=b-dk_2]$$

$$=\sum_{d\in Z_p}(1/p)(1/p^2)=1/p^2.$$

Case 4. $i_1\neq i_2$, $j_1\neq j_2$. Then

$$\Pr[y_{i_1,j_1,k_1}=a\wedge y_{i_2,j_2,k_2}=b]=\Pr[x_{i_1}+x_{j_1}\cdot k_1=a\wedge x_{i_2}+x_{j_2}\cdot k_2=b]$$

$$=\sum_{d_1,d_2\in Z_p}\Pr[x_{j_1}=d_1]\cdot\Pr[x_{j_1}=d_2]\cdot\Pr[x_{i_1}=a-d_1k_1\wedge x_{i_2}=b-d_2k_2]$$

$$=\sum_{c,d\in Z_p}(1/p)(1/p)(1/p^2)=1/p^2.$$

□

Here we get $p\binom{r}{2}$ pair-wise random variables from r fully independent random variables.

So $r=[\sqrt{2(t-1)/\epsilon\delta p}]+1$ is enough to generate at least $(t-1)/\epsilon\delta$ pair-wise independent variables.

Now we give a slightly changed algorithm employing the above idea.

Algorithm 2.

(1) Let p be a prime not less than $\sum_{j=1}^{t}|S_j|$ and add a set with $p-\sum_{j=1}^{t}|S_j|$ new elements which do not appear in any other sets, $t'\leftarrow t+1$;

(2) Let $r=[\sqrt{2(t'-1)/\epsilon\delta p}]+1$, uniformly and independently choose $x_i\in Z_p$, $i=1,2,\ldots r$, and compute $y_{i,j,k}$, $i,j=1,2,\ldots r$, $j<i$, $k=0,1,\ldots,p-1$;

(3) For all $y_{i,j,k}$ do

begin

Let x be the $\left(y_{i,j,k}-\sum_{j=1}^{l-1}|S_j|\right)$ th element of the lth set, where $\sum_{j=1}^{l-1}|S_j|\leqslant y_{i,j,k}<\sum_{j=1}^{l}|S_j|$;

Compute $\mathrm{cov}(x)=|\{j|x\in S_j\}|$;

$X_{i,j,k}\leftarrow\sum_{j=1}^{t}|S_j|/\mathrm{cov}(x)-p+\sum_{j=1}^{t}|S_j|$;

end;

(4) $X\leftarrow\dfrac{1}{\binom{r}{2}\cdot p}\sum_{i,j,k}X_{i,j,k}$

It is easy to obtain from the construction of Algorithm 2 that:

Theorem 6. Algorithm 2 is an ϵ,δ-approximation algorithm.

Proof. Similar calculation to that in the proof of Theorem 1 shows $E[X_{i,j,k}]=|\bigcup_{1\leqslant i\leqslant t}S_i|$. By Theorem 4 all $y_{i,j,k}$ are pair-wise independent. So all $X_{i,j,k}$ are pair-wise independent unbiased estimators of $|\bigcup_{1\leqslant i\leqslant t}S_i|$. By Theorem 2, estimator such as $\binom{r}{2}\cdot p>(t-1)/\epsilon\delta$ is enough to ensure that Algorithm 2 is an ϵ,δ-approximation algorithm. □

3 Analysis of the Algorithms

We now have a look at how many random bits are used in each algorithm. Let $M=\max_{1\leqslant j\leqslant t}|S_j|$, then Algorithm 1' uses

$$N(\log t+\log M)=\lceil 4t\ln(2/\delta)/\epsilon^2\rceil\cdot\log tM=O(t\log(tM)(1/\epsilon)^2\ln(1/\delta)),$$

random bits. Algorithm 2 uses

$$r\log p=([\sqrt{2(t'-1)/\epsilon\delta p}]+1)\cdot\log p=O(t^{1/2}(1/\epsilon)^{1/2}(1/\delta)^{1/2})$$

bits. For fixed ϵ and δ, as we can see here, Algorithm 2 uses much fewer random bits ($O(t^{1/2})$) than Algorithm 1' ($O(t\log tM)$).

But what about the running time bounds? We suppose all arithmetic operations can be done in $O(1)$ time. Then in Algorithm 1', the time to compute $\mathrm{cov}(x)$ is $O(t\log M)$. So we can bound its running time by $O(t\log M \cdot N)$ which is $O(t^2\log M(1/\epsilon)^2\ln(1/\delta))$.

Now consider Algorithm 2. In step 2, computing all $y_{i,j,k}$ needs $O(\binom{r}{2} \cdot p \cdot \log p)=O(t\log p(1/\epsilon)(1/\delta))$ time, considering that the computations are over Z_p. In step 3, we need $O(t)$ time to find x. To compute $\mathrm{cov}(x)$, we can first determine $c=|\{j \mid x\in S_j, j=1,\ldots,t\}|$. If $c>0$, we know that x is in some sets among S_j, $j=1,2,\ldots,t$, and not in S_{t+1}(with size $p-\sum_{j=1}^{t}|S_j|$), so $\mathrm{cov}(x)=c$. Otherwise, $\mathrm{cov}(x)=1$ because x must be in set $S_{t'}(t'=t+1)$ and not in any other set. So $O(t\log M)$ time is enough to determine $\mathrm{cov}(x)$. Thus the total running time for step 3 is $O(\binom{r}{2} p \cdot (O(t\log M)+O(t)))=O((t^2\log M+t^2)(1/\epsilon)(1/\delta))$. And the running time for step 4 is $O(\binom{r}{2} \cdot p)=O(t(1/\epsilon)(1/\delta))$. We can now conclude that the time bound of Algorithm 2 is $O((t\log p+t^2\log M)(1/\epsilon)(1/\delta))$. According to the result in number theory, there is a prime between $\sum_{j=1}^{t}|S_j|$ and $2\sum_{j=1}^{t}|S_j|$ ($\leqslant 2tM$). So we can safely bound the running time of Algorithm 2 by $O((t\cdot\log(2tM)+t^2\log M)(1/\epsilon)(1/\delta))=O(t^2\log M(1/\epsilon)(1/\delta))$. We conclude the above discussion by the following result:

Theorem 7. Algorithm 2 uses $O(t^{1/2})$ random bits and runs in time $O(t^2\log M)$ for fixed accuracy and confidence parameter ϵ,δ.

Surprisingly, Algorithm 2 doesn't necessarily cost more running time (which depends on ϵ and δ) than Algorithm 1' though it needs fewer random bits. And for fixed accuracy and confidence parameter ϵ and δ, they have the same bound $O(t^2\log M)$.

4 Further Discussions

Often, we view an ϵ,δ-approximation algorithm as a member of a family of approximation algorithms $\{A_{\epsilon,\delta}\}(\epsilon,\delta>0)$. Such a family of approximation algorithms is called a polynomial-time approximation scheme if for all ϵ and δ, the running time of $A_{\epsilon,\delta}$ is bounded by a polynomial time in n (the size of the instance), $(1/\epsilon)$ and $\ln(1/\delta)$. We note that the approximation scheme related to Algorithm 1' is a polynomial-time approximation scheme while that of Algorithm 2 is not, since $(1/\delta)$ is not bounded by any polynomial in $(\ln(1/\delta))$. Is it an inevitable result of the introduction of pair-wise independence? We leave it as an open problem for further research.

It is well-known that the technique of pair-wise independence is a powerful tool in derandomization which enables us to remove (partly) randomness from a randomized algorithm. Yet, we have not made good use of this technique enough in our algorithm because the number of the random bits used in our algorithm is not "small" if ϵ and δ are considered. So it is natural to think about using even fewer random bits in this algorithm. And, if possible, to remove all the randomness to get a deterministic algorithm. These are also good open problems.

In this paper, we take a classical combinatorial problem as an example to demonstrate the power of pair-wise independence. Then, is there any other application problem as well as theoretical problem in which this technique can be applied to remove partly the randomness inside? Any new results related to this are of great interest to the authors.

References:

[1] Karp, R. M., Pippenger, N., Sipser, M. A time-randomness tradeoff. Presented at the AMS Conference on Probabilistic Computational Complexity, Durham, NC, 1985.

[2] Koller, D., Megiddo, N. Constructing small sample spaces satisfying given constraints. In: Aggarwal, A. ed. Proceedings of the 25th ACM Annual Symposium on Theory of Computing. San Diego, California: ACM Press, 1993. 268~277.

[3] Alon, N., Goldreich, O., Hastad, J., *et al*. Simple constructions of almost k-wise independent random variables. In: Yannakakis, M. ed. Proceedings of the 31st Annual Symposium on Foundations of Computer Science. Los Alamitos, California: IEEE Computer Society Press, 1990. 544~553.

[4] Karloff, H., Mansour, Y. On construction of k-wise independent random variables. In: Goodrich, M. ed. Proceedings of the 26th ACM Annual Symposium on Theory of Computing. Montréal, Québec, Canada: ACM Press, 1994. 564~573.

[5] Wigderson, A. The amazing power of pair-wise independence. Presented at the 17th ACM Annual Symposium on Theory of Computing. Montréal, Québec, Canada, 1994. 645~647.

[6] Luby, M. A simple parallel algorithm for the maximal independent set. SIAM Journal of Computing, 1986,15(4):1036~1053.

[7] Chor, B., Goldreich, O. On the power of two-point based sampling. Journal of Complexity, 1989,5(1):96~106.

[8] Karp, R. M., Luby, M., Madras, N. Monte-Carlo approximation algorithm for enumeration problems. Journal of Algorithms, 1989,10(3):429~448.

[9] Rajeev, M., Prabhaka, R. Randomized Algorithms. Cambridge: Cambridge University Press, 1995.

并集问题的一个随机算法

张立宇, 朱 洪, 张丕兴

(复旦大学 计算机科学系,上海 200433)

摘要:随机算法由于其简洁和高效的特点正在计算中占据越来越重要的位置.但有时随机算法的优良性能并不要求用完全独立的随机变量作为它的输入.仅用成对独立的随机变量作为输入,得到了一个关于估计并集的基的问题的随机算法.这一方法可以减少随机算法中使用的随机位.对于固定的精确度ϵ和确信度δ,此算法需要$O(t^{1/2})$的随机位,比标准的随机算法所使用的随机位数$O(t\log t/M))$要少得多.而算法的执行时间并没有显著地增加$O(t^2\log M)$.

关键词:随机算法;成对独立性;并集;随机位;k-独立性

中图法分类号:O158 **文献标识码**:A