Chapter 1

Introduction

The existence of a random coin has been extensively assumed for applications such as randomizing algorithms, cryptographic protocols, and stochastic simulation experiments. The available sources of randomness are sometimes physical devices, such as Zener diodes, and Geiger counters. The sample mean of the devices as an estimate of the bias of a random coin is always rational. Thus, in this dissertation, *bias* is a rational number between 0 and $\frac{1}{2}$ unless otherwise specified. (We note that in order to determine the irrationality of the mean of a random variable is investigated by Cover [Cov73].) A *probabilistic Turing machine* (PTM) is a Turing machine with distinguished states called coin-tossing states [Gil77]. In general, it is assumed that a PTM has a fair coin. This is led by the following results: Santos showed that there is no important difference between a PTM with a fair coin and a PTM with a biased coin for the language recognition [San71, Theorem 5.1]. Moreover, any biased coin can simulate another biased coin in a constant expected time. This follows from that any biased coin can simulate a fair coin in a constant expected time [Neu51, HS70, SW84], and vice versa [KY76].

The interchangeability of the simulation among coins is measured only by the *expected* time of computation. However, it is necessary to terminate in bounded time on some applications such as realtime operations, and synchronous machines. We first consider the computation which will always terminates in *bounded* time, as opposed to *expected* time in Chapter 3. Feldman *et al.* compared these models and they noticed that there is a conceptual difference between these two kinds of measures [FIN$^+$93]. We show that the conceptual difference breaks the interchangeability of the coins. We say that a PTM *efficiently* computes when it always terminates in bounded time. We consider a PTM with a biased coin which efficiently simulates biased coins. A *coin of bias* p, denoted by $\mathcal{C}_p$, is a coin such that an outcome of the coin-tossing is "$\mathcal{H}$" with probability p, and "$\mathcal{T}$" with probability $1-p$. Let $\mathcal{S}_p$ be the set of biases efficiently simulated by a PTM with $\mathcal{C}_p$. The first result in this dissertation is the following: (i) The set $\mathcal{S}_{\frac{1}{2}}$ is equal to the set of the biases in finite figures on the binary system, (ii) the set $\mathcal{S}_{\frac{1}{3}}$ is equal to the set of the biases in finite figures on the ternary system, and (iii) the set $\mathcal{S}_{\frac{b}{a}}$ in reduced form with $a > 3$, is properly contained in the set of the biases in finite figures on the a-ary system. In this sense, $\mathcal{C}_{\frac{1}{2}}$ and $\mathcal{C}_{\frac{1}{3}}$ are more expressive coins than the others. Then, oppositely, how many least coins which efficiently simulate the set of the biases in finite figures on the a-ary system when $a > 3$? We show that the answer to this question is only *two*. The results in this chapter can be found in [Ueh95b].

In Chapter 4, we deal with a *semi-random source*, introduced by Satha and Vazirani, as an extremely general model of an imperfect source of randomness [SV84, SV86]. In a semi-random source, the previous bits generated by the source can condition the next bit in an arbitrarily

way. Accordingly, the next bit is generated by the flip of a coin whose bias is fixed by an adversary who has complete knowledge of the history of the process. The adversary is limited to choosing a bias in $[\delta, 1-\delta]$ with some positive number $0 \leq \delta \leq \frac{1}{2}$. We call a semi-random source with parameter δ a *δ-random source*. The complexity classes PP, BPP and RP are usually defined via PTMs that have access to a perfect random source as in [Gil77]. A natural question is this: Do these classes change if the PTMs only have access to an arbitrary semi-random source rather than a perfect random source? This question was first considered by Vazirani and Vazirani, who defined the class SR_p, corresponding to RP, and showed that SR_p = RP [VV85]. Later, Vazirani similarly defined the class $SBPP$, corresponding to BPP, and proved that $SBPP$ = BPP [Vaz86]. In the same manner, we introduce the class δ-PP, corresponding to PP, as follows: A language L is in δ-PP if there exists a PTM M such that (1) for $x \in L$, M accepts with probability greater than $\frac{1}{2}$ for all δ-random sources and (2) for $x \notin L$, M accepts with probability less than $\frac{1}{2}$ for all δ-random sources. We prove that δ-PP = BPP for $0 < \delta < \frac{1}{2}$. Compared with that SR_p = RP and $SBPP$ = BPP, this result is very interesting. The results for the characterization can be found in [Ueh97].

Recall that in the definition of δ-PP, the PTM M is required to work for *every* δ-random source. This also motivated us to introduce the *dual* classes $^{\exists}\delta$-RP, $^{\exists}\delta$-BPP, and $^{\exists}\delta$-PP. More precisely, we define the class $^{\exists}\delta$-PP as follows: A language L is in $^{\exists}\delta$-PP if there exists a PTM M such that (1) for $x \in L$, M accepts with probability greater than $\frac{1}{2}$ for *at least one* δ-random source and (2) for $x \notin L$, M accepts with probability less than $\frac{1}{2}$ for *all* δ-random sources. The classes $^{\exists}\delta$-BPP and $^{\exists}\delta$-RP are defined similarly. We prove that $^{\exists}\delta$-RP = NP, and $^{\exists}\delta$-BPP = $^{\exists}\delta$-PP = PSPACE for $0 < \delta < \frac{1}{2}$. These results give new characterizations of NP and PSPACE using semi-random sources.

A stochastic Turing machine (STM) is a Turing machine that can perform nondeterministic and probabilistic moves and alternate between both types. The class ABPP characterized by a polynomial time bounded STM with bounded error probability was investigated as games against nature by Papadimitriou [Pap83], Arthur-Merlin games by Babai [Bab85], and interactive proof systems with public coins [GMR85, GS86]. The class APP characterized by a polynomial time bounded STM with unbounded error probability was also investigated by Papadimitriou [Pap94]. It is well known that the classes ABPP and APP is equal to PSPACE [Sha90]. We define the class ARP using the same manner as the class RP on a polynomial time bounded STM. Comparing with the classes RP, BPP, and PP, the class ARP is naturally defined on a STM. In addition to its correspondence to the classes on a PTM, there is another reason for us to be interested in the class ARP. Condon and Ladner investigated the complexity classes defined on the interactive proof systems with some restrictions on provers [CL95]. Their results contain the complexity classes defined by a STM with a restriction on space resources or/and a restriction on the number of random bits, and characterize the complexity classes MA and NP. Liśkiewicz and Reischuk also investigated the complexity classes characterized by a STM with a restriction on space resources or/and a restriction on the number of steps [LR97]. In this sense, the class ARP can be regard as a complexity class characterized by a STM with a restriction on the acceptance probability, or the class ARP can be recognized by an interactive proof system with a restrictions on the acceptance probability of its prover. Using the characterization of NP with a semi-random source, we show that ARP = NP. This result give a new characterization of NP using a stochastic Turing machine or an interactive proof system with a restriction on the acceptance probability. The results for the characterization of the computational classes using semi-random sources can be found in [Ueh95a].

We next consider a variety of learning problem, attribute-efficient learning of Boolean concepts (or Boolean functions) in Chapter 5. This class of problems belongs to the class of exact concept

learning problems where only membership queries are allowed and is a subclass of the problems introduced by Angluin [Ang88]. A Boolean concept class F over $\{0,1\}^n$ is a set of Boolean concepts $f \subseteq \{0,1\}^n$. Since we identify f with its characteristic function $\chi_f : \{0,1\}^n \to \{0,1\}$, we treat the concept f as a Boolean function. The learner tries to find out which function $f \in F$ is chosen by a teacher, an adversary, or by chance. For this reason the learner may ask queries $a \in \{0,1\}^n$ and the teacher answers with the correct value $f(a)$. The queries can be chosen adaptively, i.e., the i-th query may depend on the answers to the first $i-1$ queries. After having asked $a_1, \ldots, a_i$, a concept g is consistent with f if $f(a_1) = g(a_1), \ldots, f(a_i) = g(a_i)$. The learner has finished the job if a unique concept $f \in F$ is consistent with the given answers. Then f has been "learned". All queries have the same cost. The cost of a learning strategy for $f \in F$ is the number of queries to learn f. The cost of a learning strategy for the concept class F is the maximum of the cost to learn f for all $f \in F$ with the considered learning strategy. The task is to design learning strategies whose cost is close to known lower bounds for the learning problem. Since we ask binary queries, the simplest lower bound is the information theoretic lower bound $\lceil \log |F| \rceil$. For many of our problems this lower bound is almost optimal. Sometimes we prove better lower bounds by adversary arguments.

In the following we describe the concept classes we are interested in. These complexity classes typically are much smaller than the class of all Boolean functions. E.g., Angluin, Hellerstein, and Karpinski [AHK93] have investigated the class of functions representable by read-once formulas. We consider classes F based on one Boolean function g (which not necessarily has to belong to F). For some sets $S \subseteq \{1, \ldots, n\}$ (typically all sets or all sets of cardinality k) there exists one function $g_S \in F$ that is a subfunction of g where each x_i with $i \notin S$, say nonessential variable, is replaced by some given constant a_i. The identification of the unknown function in F is equivalent to the identification of the set of essential attributes (sometimes also called relevant variables). The situation described above is known as attribute-efficient learning. Littlestone [Lit88] obtains a lot of results, among them optimal results for disjunction in the mistake-bound model and Bshouty and Hellerstein [BH96] study also different models of learning.

We obtain more precise results for the attribute-efficient learning of special classes of functions. We consider the classes based on one of the following Boolean functions; disjunction, parity, and threshold. They are the most important basic gates in circuit theory and, therefore, the most fundamental functions. This fact motivates us to consider these special cases of attribute-efficient learning. In applications, we obtain the considered problem in the following case; if we know that some wires into a gate are destroyed and we like to identify which wires are destroyed by testing the gates on chosen inputs. This application motivates also the investigation of large k, e. g., $k = n - O(1)$. Interestingly, the complexity of the problem is completely different depending on the kind of the function when k is large.

Our results have been influenced by more general problem classes. Concept learning with membership queries is a special problem of the class of problems which have been considered as so-called "search problems" by Picard [Pic65] and Ahlswede and Wegener [AW87]. A complete learning strategy can be described as a binary decision tree where the inner nodes describe the queries and the outgoing 0-edge (resp. 1-edge) leads to the next query if the last query has been answered by 0 (resp. 1). At the leaves we find the function which is the chosen concept. The cost of the learning strategy is equal to the depth of the corresponding binary decision tree. Hence, lower bound techniques can be adopted from the theory on binary decision trees (see [Weg87]). A preliminary version of the research of attribute-efficient learning appears as an extended abstract in [UTW97].

We next turn to consider parallel complexity of maximality problems in Chapter 6. Since Karp and Wigderson showed that the maximal independent set (MIS) problem is in the class

NC [KW85], much work has been devoted to the study of parallel complexity of maximality problems. A typical maximality problem on graphs is to find either a maximal vertex-induced subgraph (MVIS) or a maximal edge-induced subgraph (MEIS) that satisfies a specified graph property. A common feature of the MVIS or MEIS problems is that they can be solved by simple greedy algorithms but are very difficult from the parallel point of view. So far, only a few MVIS or MEIS problems have been shown to be in NC or RNC [And85, Che91, Che96, Che95b, DGL91, IS86, KR91, PV93, SM95]. Parallel complexity of many natural MVIS or MEIS problems remains unknown.

The *maximal path set* (MPS) problem is to find, given an undirected graph $G = (V, E)$, a maximal subset F of E such that the subgraph induced by F is a forest in which each connected component is a path. The MPS problem is equivalent to the MEIS problem associated with the property "acyclic and maximum-degree ≤ 2". In [Che96], Chen introduced this problem and showed that parallel algorithms for it can be used to design parallel approximation algorithms for the famous shortest superstring problem. It is worth mentioning that the latter problem has been extensively studied due to its important applications in DNA sequencing and data compression [BJL+91, CGPR94, KPS94, TY93]. In [Che96], Chen presented an NC algorithm and an RNC algorithm for the MPS problem. The former runs in $O(\log^3 n)$ time with $O(n+m)$ processors on a CRCW PRAM and the latter runs in $O(\log^2 n)$ expected time with $O(n+m)$ processors on a CRCW PRAM. In this dissertation, we present two faster parallel algorithms for the problem. Our RNC algorithm runs in $O(\log n)$ expected time with $O(n+m)$ processors on a CRCW PRAM. This algorithm is faster and more efficient than Chen's RNC algorithm. Our NC algorithm for the MPS problem is obtained by *carefully* derandomizing the RNC algorithm, and it runs in $O(\log^2 n)$ time with $O(\Delta^2(n+m)/\log n)$ processors on an EREW PRAM, where Δ is the maximum degree of the input graph. Compared with Chen's NC algorithm, this algorithm is faster, runs on a weaker computation model, and is more efficient for input graphs of bounded degree. An immediate consequence of the results is that the parallel approximation algorithms for the shortest superstring problem given in [Che96] can be made faster. These parallel algorithms for the MPS problem was investigated in [UCH96, UCH97].

In [JLD92], Jiang *et al.* introduced an interesting variation of SSP. Let $S = \{s_1, \cdots, s_n\}$ be a set of n strings over an alphabet Σ. A *superstring-with-flipping* of S is a string s over Σ such that for each i, at least one of s_i and its reverse is a substring of s. The *amount of compression* achieved by a superstring-with-flipping s of S is $|S| - |s|$, where $|s|$ is the length of s and $|S|$ is the total length of the strings in S. Define SSP_F to be the following problem: Given a set S of strings, find a shortest superstring-with-flipping of S. Like SSP, SSP_F is NP-hard [JLD92] and it is of interest to design approximation algorithms for SSP_F. In [JLD92], Jiang *et al.* presented a polynomial-time approximation algorithm for SSP_F that produces a superstring-with-flipping whose length is at most 3 optimal. They also pointed out that there is a greedy algorithm for SSP_F that produces a superstring-with-flipping by which the amount of compression achieved is at least $\frac{1}{2}$ optimal [JLD92]. At present, no better sequential approximation algorithms for SSP_F are known. Also, no parallel approximation algorithm for SSP_F has been given previously. Here, using the ideas in our parallel algorithms for the MPS problem, we give an NC approximation algorithm for SSP_F that produces a superstring-with-flipping by which the amount of compression achieved is at least $\frac{1}{3+\epsilon}$ optimal for any $\epsilon > 0$. This NC approximation algorithm was presented in [UCH96].

We also consider the *vertex* counterpart of the MPS problem, which is the MVIS problem associated with the property "acyclic and maximum-degree ≤ 2". More precisely, we consider the following problem: Given an undirected graph $G = (V, E)$, find a maximal subset U of V such that the subgraph induced by U is a forest in which each connected component is a path. We call this problem the *maximal linear forest* (MLF) problem. For several special cases of the

MVIS problem associated with the property "acyclic", which is equivalent to the problem of computing a *maximal forest*, NC algorithms are given in [CH95]. However, parallel complexity of this problem still remains open. In this dissertation, we show that a related problem, namely, the MLF problem is in NC. This result can be found in [UC97]. In addition to its correspondence to the problem of computing a maximal forest, there is another reason for us to be interested in parallel complexity of the MLF problem. The MPS problem can be efficiently reduced to the MLF problem as follows: To solve the MPS problem for a given graph G, it suffices to solve the MLF problem for the edge graph of G. Thus, efficient parallel algorithms for the MLF problem can be used to design parallel approximation algorithms for the shortest superstring problem, too. Surprisingly, designing an NC algorithm for the MLF problem is a little more difficult than for the MPS problem even though they look very similar. Our RNC algorithm for the MLF problem runs in $O(\log n)$ expected time using n^2 processors on a CRCW PRAM. Our NC algorithm for the MLF problem is obtained by derandomizing the RNC algorithm. It runs in $O(\log^2 n)$ time using n^4 processors on an EREW PRAM.

Our RNC algorithm for the MPS problem has a similar structure to that of Israeli and Itai's RNC algorithm (I&I algorithm, for short) for the maximal matching problem [II86]. Our RNC algorithm for the MLF problem also has a similar structure to that of Luby's RNC algorithm for the MIS problem [Lub86]. All of these algorithms proceeds in stages. In each stage, the main jobs of the I&I algorithm and our algorithm for the MPS problem are to compute a random matching M in a certain subgraph of G and to delete (from G) some edges incident to the vertices matched by M. On the other hand, in each stage, the main jobs of Luby's algorithm and our algorithm for the MLF problem are to compute a certain independent set I in a certain (augmented) subgraph of G and to delete (from G) the vertices in I and (part of) their neighbors. In both the I&I algorithm and Luby's algorithm, the expected number of edges deleted in each stage is a constant fraction of the number of edges in G [II86, Lub86]. However, both our algorithms do not have this property. Instead, we define a *potential function* and prove that in each stage, the value of the potential function decreases by a constant fraction *on average*. This is the key for us to show that the algorithms run very fast.

The NP-completeness of several problems are shown in the last chapter of this dissertation. To show the NP-completeness of the problems on a simple structure means from two points of view. First, it is easy to consider the approximation/probabilistic algorithm for the problems on a simple structure. Secondly, since the reduction to the other problems is easy to consider, it is useful to analyze the more complex problems.

The first NP-complete problem is the generalized popular puzzle known as Hi-Q [UI90]. Complexity of problems concerning various generalized puzzles and games have been studied [ET76, Sch78, SC79]. Most of these generalizations were, however, artificial rather than natural, i.e., they were games on propositional formulas, sets, or graphs, or sometimes played three dimensionally. Complexity results on generalized popular two-person games such as Chess, Checkers, Go, Shogi, Gomoku, and Othello have been presented [Rei80, FL81, Rob83, Rob84, AKI87, IK94]. These generalization is natural in the sense that the board is extended to the size $n \times n$ with many pieces. Few results, however, have been presented concerning popular puzzles, which are naturally generalized. We consider a puzzle commonly known as Hi-Q. The objective of the puzzle is to make the board to a position with just one token on the goal. Thus a sequence of jumps which leads the board consisting of no tokens but on the goal will be an answer for the puzzle. The puzzle is generalized in such a way that an initial position is given on the extended board of size $n \times n$, and a goal is also given on which only one token will finally be placed. We show that the problem to determine whether there is an answer for a given generalized Hi-Q is NP-complete. The NP-hardness can be obtained by reducing from a variation of the

Hamiltonian cycle problem.

The second NP-complete problem is the vertex cover (VC) problem (or the maximum independent set problem) on a cubic, planar, and 3-connected graph of girth greater than 3. It is well known that finding a VC, or a maximum independent set of a given graph is NP-complete even for a cubic planar graph [GJ79]. We add the 3-connectedness and the restriction on the girth to the previously known result. The result simplifies the proof of NP-completeness of several well-known computational geometry problems involving 3-dimensional convex polyhedra investigated in [DG95], and also simplifies the proof of NP-completeness of computing the asteroidal number of a graph investigated in [KKM96]. Using the fact for the 3-connectedness of the cubic planar graph, we also have two NP-complete problems on a triangulation of the sphere, which is widely investigated and has many applications [PS90].

The rest of the dissertation is organized as follows. In Chapter 2 we introduce some notations and state basic results. In Chapter 3, the interchangeability of the biased coins on the efficient computation model is shown. In Chapter 4, we consider the complexity classes characterized by a semi-random sources. In Chapter 5, the algorithms for attribute-efficient learning of the special classes are designed. In Chapter 6, the parallel algorithms for the MPS and MLF problem are designed. The NP-completeness of several problems are shown in Chapter 7.