It is not possible to wait for inspiration, and even inspiration alone is not sufficient. Work and more work is necessary. Man blessed by genius can create nothing really great, not even anything mediocre, if he does not toil as hard as a slave.

Piotr Ilyich Tschaikovsky

5.1 Objectives

This chapter is devoted to two paradigms of the design of randomized algorithms, namely the amplification of success probability by repeating runs on the same input and random sampling. The reasons for presenting both these methods in one chapter are their similarity and their equally balanced combination in several applications. Thus, for some randomized algorithms, it is not possible to determine which of these two methods is primarily responsible for success.

In Chapter 2 we called attention to the fact that amplification is a method for reducing the error probability below an arbitrarily given small constant $\epsilon > 0$. We underlined the importance of this observation by classifying randomized algorithms with respect to the speed of error probability reduction with the number of computation repetitions on the same input. In this chapter we aim to present algorithms for which amplification does not only increase the success probability, but directly stamps the process of the algorithm design. Moreover, we do not want only to follow the naive approach of repeating the whole computation on the same input, but also to introduce a more advanced technique that prefers to repeat only some computation parts or to repeat different parts differently many times. The idea is to pay more attention to computation parts in which the probability of making mistakes is greater than in other ones.

With random sampling we aim to document the power of this method by designing efficient randomized algorithms solving problems for which no deterministic polynomial-time algorithm has up to now been discovered.\footnote{and maybe for which no efficient deterministic algorithms exist at all}

This chapter is organized as follows. Section 5.2 introduces the above mentioned generalized version of the amplification method. This method is used...
to design randomized algorithms solving the minimum cut problem for multigraphs. In Section 5.3 we combine amplification with random sampling in order to design a practicable one-sided-error Monte Carlo algorithm for the well known, NP-hard 3-satisfiability (3SAT) problem. Though this algorithm runs in exponential time, it is much faster than algorithms running in $O(2^n)$ time and it can be successfully applied for relatively large instances of 3SAT. In Section 5.3 we present an application of random sampling that results in a Las Vegas polynomial-time algorithm for a number-theoretic problem, which is not known to be in P. Hence, this randomized algorithm is the only efficient way known for solving this problem. Altogether this chapter presents impressive examples documenting the superiority of randomized algorithms over their best known deterministic counterparts. As usual, we finish the chapter by summarizing the most important ideas and results presented.

5.2 Efficient Amplification by Repeating Critical Computation Parts

The aim of this section is to introduce the method of amplification of the success probability as a method for the design of randomized algorithms, and not only (as considered until now) as a technique for error probability reduction of algorithms already designed. For this purpose we consider the following minimization problem MIN-CUT.

MIN-CUT

*Input:* A multigraph $G = (V, E, c)$, where $c : E \rightarrow \mathbb{N} - \{0\}$ determines the multiplicity of the edges of $G$.

*Constraints:* The set of all feasible solutions for $G$ is the set

$$\mathcal{M}(G) = \{(V_1, V_2) \mid V_1 \cup V_2 = V, V_1 \cap V_2 = \emptyset\}$$

of all cuts of $G$.

*Costs:* For every cut $(V_1, V_2) \in \mathcal{M}(G)$,

$$\text{cost}((V_1, V_2), G) = \sum_{e \in S(V_1, V_2)} c(e),$$

where $S(V_1, V_2) = \{\{x, y\} \in E \mid x \in V_1 \text{ and } y \in V_2\}$

{i.e., cost($(V_1, V_2), G)$ is equal to the number of edges between $V_1$ and $V_2$}

*Goal:* minimum

The best known deterministic algorithm for MIN-CUT runs in time

$$O\left(|V| \cdot |E| \cdot \log \left(\frac{|V|^2}{|E|}\right)\right),$$