

Preface

The wisest of the wise may err.

Aeschylus, Fragments

The study of the effect that errors may have on computation started at the very beginning of the history of computer science. It dates back to the early 1950s, to the research by von Neumann¹ and Moore-Shannon² on the design of reliable Boolean networks built with low-reliability components. Since then the study of fault-tolerant computing has evolved into a broad discipline, one that encompasses all aspects of reliable computer design: from the identification of failure dynamics in integrated circuits to the design of robust software.

The design of reliable computers is much harder than the design of other complex human-made objects. Citing the *IEEE Spectrum*,³

[...] Information-processing errors can occur through a failure lasting a billionth of a second in one of the hundreds of thousands of digital components that switch billions of times a day.

In fact, it seems rather hopeless to attain high reliability in computer design by the so-called fault-avoidance techniques, that is, by only relying upon high-quality thoroughly tested components. Owing to the intrinsic complexity of modern computers, it is more sensible to focus on software robustness for the detection and/or correction of errors which will invariably occur as information is being stored, transferred, and manipulated. This is typically achieved by introducing some redundancy (in the information) to improve the reliability of the computational systems, as already suggested in the seminal papers by von Neumann and Moore-Shannon.

¹*Automata Studies*, Princeton University Press (1956).

²J. Franklin Inst. (1956).

³*IEEE Spectrum*. Oct. 81, p. 41.

One more reason to prefer fault-tolerance to fault-avoidance techniques is given by the error statistics in the electronic systems. Typically, failure occurrence in electronic components follows the so-called *bathtub* curve. This means that a large failure rate is more likely for a short initial period known as the burn-in period. After this, components may experience a small failure rate for the rest of their operational life. Therefore a small probability of component failure exists during the useful life. It seems natural to look at error-correcting and error-detecting coding techniques as the most promising means to provide low-cost error control in computation.

The original scope of the theory of error-correcting codes was the design of reliable communication systems. It is clear that the problem of reliable computation differs significantly from the problem of reliable communication: for example, communication error-control schemes usually assume perfectly reliable computing and processing at the transmitter and receiver ends. They put fewer severe restraints on computation time for error correction and obey different statistics for error occurrence than ruling computer systems. Nevertheless, the fundamental principles of communication coding theory also are essential to the understanding and design of error control for reliable computation. Winograd and Cowan⁴ pointed out that (subject to some assumptions) Shannon's noisy-channel coding theorem may be extended to include noisy computation. Thus, computation of logical functions in the presence of unreliable modules may be thought of as analogous to communication over a noisy channel of a certain capacity.

In this book we shall be mainly concerned with fault tolerance in the context of algorithmic search theory. Problems of search, with their wide applicability, allow us to show fault-tolerant techniques as they apply to many different contexts. Moreover, search theory is one of the classical fields in the science of the computation.

As with fault-tolerant techniques, the necessity of efficient search procedures arose very early in computer science. One can say that the theory of searching starts with the first computers.

In the early 1950s, while the first fault-tolerant techniques were being devised, the new availability of larger and larger random-access memories eventually led to the recognition that *searching* was an interesting problem in its own right. All of a sudden memory space was not anymore a problem of scarcity but rather because of the lack of efficient techniques for making the best use of it. Almost instantaneously, *searching* became a flourishing field of research. The first surveys on search problems had already been published in 1956.⁵

Since these pioneering works, searching has been a very active field of investigation. As a matter of fact, search procedures can constitute the most time-consuming parts of software, and the substitution of a good search method for a bad one then brings a substantial improvement. The entire third volume of Donald Knuth's celebrated series *The Art of Computer Programming* is focused on search and

⁴*Reliable Computation in the Presence of Noise* (1963).

⁵See [29, 96, 170].

sorting algorithms. In the introduction of this volume, Knuth makes the following claim:

Indeed, I believe that virtually every important aspect of programming arises somewhere in the context of sorting and searching.

Whether every scientific problem can be reduced to a search problem may be a matter of philosophical debate. Nonetheless, it is undeniable that many problems of practical importance can be reduced to searching.

The structure of a typical search problem appears more often than one would expect, in surprisingly diverse scenarios. A search problem can be the task of: (1) identifying objects within a set via a series of tests (identification); (2) learning the structure of a given function via sampling (learning); (3) determining the most concise way of representing a given piece of information so that recovery is uniquely achievable (encoding); (4) distinguishing patterns on the basis of exact or approximate examples (classification). Depending on the application at hand, any one of the above may be the most appropriate or convenient problem formulation. However, a unified perspective has the advantage that it allows for cross-fertilization of ideas and methodologies.

Our motivation for studying fault-tolerant computation in the context of search is twofold: From a more analytical point of view, there exists a large class of combinatorial problems of practical importance which can be studied in a unified way under the common concept of *search*. Moreover, from a more application-oriented perspective, *search problems* in their many different facets lie at the heart of managing large data sets, one of today's major IT challenges. Current computer applications handle data sets that are not only larger than ever before but also include highly complex objects. To tame the data deluge, it is crucial to develop the capacity for appropriately organizing, storing, and, most importantly, *searching* through such a mass of data.

In our excursus on fault-tolerant search algorithms, we shall see that many methodologies originally thought of and developed in the context of search problems were proved useful, in some cases even rediscovered, in areas that might appear far from searching. To disclose such connections and bridge these gaps among connected fields of research will be another aim of this book.

The book is self-contained and assumes no special knowledge beyond a standard undergraduate background in basic algorithmics, complexity, combinatorics, and probability theory. All necessary concepts are introduced and all the results are proved before they are used, even when one could assume them known. There are a very limited number of exceptions, where we decided to either omit a detailed involved proof or delay its presentation with respect to its use, because we preferred to focus on the broad picture without risking to distract the reader from the main train of thoughts. Even in these cases, all necessary pointers are always provided. The bibliographic notes refer the reader to appropriate sources of information for closing any gap of knowledge necessary for a complete understanding of the proof.

Each chapter is concluded by a set of exercises of varying difficulty. In general, these exercises are meant to present problems which should allow the reader to

deepen his/her understanding of the material covered in a chapter. Some of these exercises are variants, extensions, or particular cases of proofs or algorithms presented in the text. Some exercises are also used to cover important complementary results which should have been but eventually were not included due to logistic constraints.

At the end of each chapter a brief historical account of some fundamental publications has been included, describing the origins of the main results. The bibliography at the end of the book includes most of the key articles in the field and references to other books and survey papers on the subject.

Salerno/Verona, Italy
September 2012

Ferdinando Cicalese

Fault-Tolerant Search Algorithms

Reliable Computation with Unreliable Information

Cicalese, F.

2013, XV, 207 p. 14 illus., Hardcover

ISBN: 978-3-642-17326-4