ACM MONOGRAPH SERIES

Decomposability

QUEUEING AND COMPUTER SYSTEM APPLICATIONS

P. J. Courtois

DECOMPOSABILITY

Queueing and Computer System Applications

P. J. Courtois

MBLE RESEARCH LABORATORY
BRUSSELS, BELGIUM

ACADEMIC PRESS New York San Francisco London 1977

A Subsidiary of Harcourt Brace Jovanovich, Publishers

ACADEMIC PRESS, INC.
111 Fifth Avenue, New York, New York 10003

United Kingdom Edition published by
ACADEMIC PRESS, INC. (LONDON) LTD.
24/28 Oval Road, London NW1

Library of Congress Cataloging in Publication Data

Courtois, P J
Decomposability.

(ACM monograph series)
Bibliography: p.
Includes index.
1. Decomposition method. 2. Electronic digital computers–Evaluation. 3. Queuing theory. I. Title. II. Series: Association for Computing Machinery. ACM monograph series.
QA402.2.C68 519.8′2 76-19484
ISBN 0–12–193750–X

PRINTED IN THE UNITED STATES OF AMERICA

DECOMPOSABILITY

QUEUEING AND COMPUTER SYSTEM APPLICATIONS

ACM MONOGRAPH SERIES

Published under the auspices of the Association for Computing Machinery Inc.

Editor ROBERT L. ASHENHURST *The University of Chicago*

A. FINERMAN (Ed.) University Education in Computing Science, 1968
A. GINZBURG Algebraic Theory of Automata, 1968
E. F. CODD Cellular Automata, 1968
G. ERNST AND A. NEWELL GPS: A Case Study in Generality and Problem Solving, 1969
M. A. GAVRILOV AND A. D. ZAKREVSKII (Eds.) LYaPAS: A Programming Language for Logic and Coding Algorithms, 1969
THEODOR D. STERLING, EDGAR A. BERING, JR., SEYMOUR V. POLLACK, AND HERBERT VAUGHAN, JR. (Eds.) Visual Prosthesis: The Interdisciplinary Dialogue, 1971
JOHN R. RICE (Ed.) Mathematical Software, 1971
ELLIOTT I. ORGANICK Computer System Organization: The B5700/B6700 Series, 1973
NEIL D. JONES Computability Theory: An Introduction, 1973
ARTO SALOMAA Formal Languages, 1973
HARVEY ABRAMSON Theory and Application of a Bottom-Up Syntax-Directed Translator, 1973
GLEN G. LANGDON, JR. Logic Design: A Review of Theory and Practice, 1974
MONROE NEWBORN Computer Chess, 1975
ASHOK K. AGRAWALA AND TOMLINSON G. RAUSCHER Foundation of Microprogramming: Architecture, Software, and Applications, 1975
P. J. COURTOIS Decomposability: Queueing and Computer System Applications, 1977

In preparation

ANITA K. JONES (Ed.) Perspectives on Computer Science: From the 10th Anniversary Symposium at the Computer Science Department, Carnegie-Mellon University
JOHN R. METZNER AND BRUCE H. BARNES Decision Table Languages and Systems

Previously published and available from The Macmillan Company, New York City

V. KRYLOV Approximate Calculation of Integrals (Translated by A. H. Stroud), 1962

Foreword

Dr. Courtois has produced an important book on the use of computers to study computers. His book is about computation: analytic and numerical methods for examining the behavior of complex systems. At the same time it is about computers: its area of application is the design of computers and the analysis of their performance.

This kind of incestuous or introspective relation of computers to their own kind is not at all unusual—and for a very good reason. Much scientific activity today, in almost all fields of inquiry, is directed toward understanding complexity—the complexity of thunderstorms and ecological systems, the complexity of genetic control of developing organisms, the complexity of human thought processes, and the complexity of computers. Since the study of complex systems calls for new experimental methods and new computational tools, a substantial part of the effort directed toward the study of complexity has been aimed at the development of such tools. In the past twenty-five years, we have seen a great flowering of computational mathematics, providing us with linear, integer, and dynamic programming, with queueing theory, with simulation techniques, and with great advances in the classical methods of numerical analysis.

The computer has played a central role in spurring these developments, for computation with an electronic computer is a very different matter from computation with an old-fashioned desk calculator, calling for new approaches and new points of view, and creating new aspirations as well. The time is long past, of course, since anyone has imagined that computer brute force could substitute for skillful analysis, and thoughtful analysts never held the illusion that such a substitution was possible. Nature has no difficulty whatsoever in inventing systems whose exact analysis defies the most powerful efforts of present or prospective computers—and large computers themselves constitute one class of such systems.

What we cannot do by brute force we must do with cunning. One form of cunning is to recognize that the systems produced by Nature and by the art of man are seldom as complex as they might be, or as their size and number of components might lead us to expect them to be. Fortunately for our prospects of understanding complex systems, it is not common for each element in such systems to interact strongly with each other element. On the contrary, the world, even the world of complexity, is mostly empty. The matrices that describe complex systems tend to be sparse matrices, and very unrandom ones. We can augment our analytic power greatly by exploiting this sparsity and this order.

Dr. Courtois' analysis is based on his important observation that large computing systems can usefully be regarded as nearly completely decomposable systems—systems arranged in a hierarchy of components and subcomponents, with interactions within components that are strong and fast compared with the interactions between components at the same level. Near-decomposability is not peculiar to computing systems: it has been observed in economic structures and in a variety of biological models, genetic and developmental. Only quite recently have its computational implications begun to be explored.

By exploiting with great originality the near-complete decomposability of large computer structures, Dr. Courtois has greatly advanced our ability to compute their behavior, and hence to design them. He has made an important conceptual contribution which is, at the same time, an important practical one. Both hardware and software designers will find in this book a set of powerful methods for systems analysis, as well as a clear exposition of the theory of nearly completely decomposable systems upon which these specific analytic methods are based. While the book addresses itself specifically to computer applications, it should also have substantial interest for scientists investigating other kinds of complex systems, who may find these or similar techniques applicable to the solution of their own computational problems.

Carnegie-Mellon University HERBERT A. SIMON
July 1976

Preface

This monograph groups the results of research work started seven years ago. The intention was to progress toward analysis methods that would be more effective in the evaluation of computer system performance, and which should ultimately lead to the design of optimized systems.

Due to the ever-increasing complexity of computer systems and of their applications, this research found itself quickly and naturally oriented toward methods proceeding by decomposition and approximations; the separation of problems and the conceding of approximations have always been golden rules for tackling the complex reality around us. We first started rather empirically using successive approximations that exploited the great differences of magnitude of the speeds at which information flows at the various levels of a computer memory hierarchy (Courtois and Georges, 1970). But it was the discovery of the aggregation methods widely used in economics and of the results obtained by H. A. Simon and A. Ando (1961) on nearly completely decomposable structures of linear models that made our work take a very decisive turn. It permitted us to justify more rigorously assumptions that we had previously envisaged but empirically, and it enlarged our investigation ground considerably.

The book may be divided into three parts. The first part, being comprised of the first three chapters, is of rather general interest. It aims at gathering together some basic elements of a theory of nearly completely decomposable stochastic matrices. The Simon–Ando theorems are presented, some of their aspects being dealt with in precise detail. Elements of Wilkinson's (1965) perturbation theory inspired us to undertake also a study of the sharpness of the accuracy of the Simon–Ando approximation. This study leads to the definition of criteria of near-complete decomposability.

The second part, which is composed of the next three chapters, is devoted to the analysis of stochastic queueing networks which appear as a type of

key model in our work. On the one hand, these models admit, at least in their simplest forms, a matrix representation that allows the theory of decomposability to be exploited straightforwardly; on the other hand, congestion problems in information processing systems may, in many respects, be adequately studied by means of queueing networks models. A method of analysis by decomposition and aggregation for these models is proposed and discussed at length.

The last part of the book deals concretely with the problem of computer system performance evaluation. The material of the first two parts is applied to the analysis of different aspects, hardware and software, of the dynamic behavior of computer systems and user programs. Chapter VIII is a shortened version of a study of program paging behavior made in collaboration with H. Vantilborgh (Courtois and Vantilborgh, 1976); Chapter IX gives a detailed analysis of an aggregative model of a typical multiprogramming time-sharing computing system. It is hoped this last part should illustrate that aggregation is not only an efficient technique for obtaining quantitative results but also for gaining insight and conceptual clarity on the parts played by the many parameters of a complex model.

Finally, a last chapter examines the striking affinity that appears to exist between the concept of aggregate in nearly completely decomposable structures and the notions of module and level of abstraction so frequently invoked in computer system design and software engineering. Variable aggregation appears in this context as a technique that could enable the level by level evaluation of a system to proceed in pace with its level by level design.

In the main, the chief result of this work is that it proposes a rather general approach to computer system model building and suggests a framework in which it seems possible to coordinate future analyses carried out in various directions and at different levels of detail.

Acknowledgments

This book would not have been written without the support of the MBLE Research Laboratory, Brussels, and of its director, Professor V. Belevitch, to whom go my foremost thanks. I should like also to thank D. L. Parnas, who contributed many discussions to the material of Chapter X. Moreover, his invitation to the Computer Science Department of Carnegie–Mellon University during the winter 1971–1972 led to personal contacts with Professor H. A. Simon to whom I owe a great debt of thanks. His advice and encouragement were invaluable for the work (Courtois, 1972) done at Carnegie–Mellon which was already the substance of this monograph.

I am also greatly indebted to my friends of the MBLE Research Laboratory for their help. The work benefited from the early assistance of J. Georges, while H. Vantilborgh followed the manuscript through several changes and is at the origin of numerous improvements; both of them, together with M.-J. Van Camp prepared the computer programs which yielded the numerical data exploited in Chapters XIII and IX, and in Appendix III. In addition I would like to thank Professors J. Meinguet, G. de Ghellinck, G. Louchard, and E. Milgrom who served on the committee for my doctoral dissertation at the Catholic University of Louvain, Belgium, for the useful advice and comments they gave on this occasion.

Finally, my thanks are due to Mrs. A. Toubeau for her patient and expert typing of the manuscript, and to C. Semaille who realized the figures.

Contents

Introduction and Overview

> The concept of hierarchic order occupies a central place in this book, and lest the reader should think that I am riding a private hobby-horse, let me reassure him that this concept has a long and respectable ancestry. So much so, that defenders of orthodoxy are inclined to dismiss it as "old hat"—and often in the same breath to deny its validity. Yet I hope to show as we go along that this old hat, handled with some affection, can produce lively rabbits.[1]
>
> A. KOESTLER (1967)
> *The Ghost in the Machine*

The research work reported in the following pages was inspired by the desire of finding adequate methods to evaluate and predict the performances of current general-purpose computer systems.

Fundamentally this problem amounts to constructing appropriate models of the dynamic behavior of these systems. By "model" we mean a set of relations between unknowns (typically, measures of efficiency, such as the system response time or the resource utilization factors) and various parameters that represent the relevant characteristics of the system and of its work load. These relations must be sufficiently simple so as to permit the evaluation of the unknowns; they must at the same time be faithful to the system, i.e., they must capture the relevant laws that govern its mode of operation. Rather obviously, such models with useful and accurate predictive properties cannot be constructed if the system is not first properly understood. To solve the problem of computer performances prediction requires not only powerful evaluation techniques but also a better understanding of the basic principles

[1] Reprinted with the permission of A. D. Peters & Co. Ltd., London.

of system behavior. These two types of requirements define the main lines of conduct of our work.

Analyses of computer system behavior have abounded during the past decade; surveys by Lucas (1971) and Graham (1973), among others, give evidence of this efflorescence. Many of these analyses, however, focus their attention on a particular aspect of system behavior; they analyze a processor-sharing mechanism, a store management strategy, a class of page replacement algorithms, an access method to rotating storage devices, a job scheduling policy, etc. Such investigations are usually carried out in depth, by means of well-defined but simplified models in the context of which appropriate mathematical disciplines can be exploited. They provide a body of knowledge that, to a certain extent, helps to understand how each of these mechanisms operates and compares in efficiency with competitors. These analytical studies made in isolation are, in a way, a normal step in engineering; "scientific progress has been made by analysis and artificial isolation . . .," as Russell (1948) observed.

Nevertheless, these isolated analyses give practically no information on the way the various mechanisms cooperate and interact upon each other within the wholeness of a single system. The influence of a given component upon the behavior of a complete installation, the contribution of a part to the efficiency of the whole are beyond the scope of these investigations. It would be only a lesser evil if computing systems enjoyed what von Bertalanffy (1968) calls the "summativity" character; then, as "a heap of heaps" or a parallelogram of mechanical forces, the variations in the behavior of the total system would merely be the sum of the variations of its elements considered in isolation. But the behavior of a computing system cannot be summed up from its isolated parts; the behavior of the component parts can be quite different within the system from what it is in isolation. The high degree of resource sharing, multiprocessing, and exploitation schemes such as multiprogramming and multiaccess have introduced complex dependencies between the various processes that take place in a modern computing system. In the face of this complexity, the isolated analyses mentioned above leave the designer without means to assess the global consequences of his particular design options. As Simon (1969) put it:

> Only fragments of theory are available to guide the design of a time-sharing system or to predict how a system of a specified design will actually behave in an environment of users who place their several demands upon it. Most actual designs have turned out initially to exhibit serious deficiencies; and most predictions of performance have been startlingly inaccurate.
>
> Under these circumstances, the main route open to the development and improvement of time-sharing systems is to build them and see how they behave. And this is what has been done. They have been built, modified, and improved in successive stages. Perhaps theory could have anticipated these experiments and made them unnecessary. In fact, it didn't; and I don't know anyone intimately acquainted with these exceedingly

complex systems who has very specific ideas as to how it might have done so. To understand them, the systems had to be constructed, and their behavior observed.

Comprehensive analyses of complete systems are of course being attempted to alleviate this state of affairs. These attempts in general resort to simulation techniques. While analytic modeling is limited by the lack of algebraic solutions for complex models, simulation is in principle apt to investigate systems of arbitrary complexity. This generality is, alas, achieved at some expense. The results obtained by simulation are often difficult to interpret with a reasonable degree of confidence. In many cases, above a certain level of complexity, the validity of a simulation model can only be asserted by checking that the model is an exact copy of the real system; as a consequence of this, the analyst is inclined to define and build his model at a level of detail that tends to approach the complexity of the real system. Such detailed models rapidly become difficult to understand and give per se less and less insight into the real system behavior. They yield results that are received with much skepticism; moreover, they quickly become difficult to modify and expensive to adjust to different design alternatives.

The moral is that existing modeling approaches have their own advantages and limitations; and that in face of the different structures of the components of a computing system, and especially of the many different levels of details at which performance prediction needs to take place, no single modeling technique simply asserts itself as being always the most useful. On the contrary, these techniques appear to complement each other.

There are two possible attitudes to take if we want to improve upon this situation: the search for a new evaluation technique more powerful and more general than existing ones, or the search for a general framework in which the different tools of analysis that are already available could be integrated. We opted for this second and less ambitious approach which appeared more promising.

From what has been said, one can already have an inkling of what this second approach may require. We need:

(1) *Criteria* according to which a computing system could be dissected into constituents that can be understood, analyzed, and calibrated separately. Once this is done, our task will be simplified because (a) the model of a constituent is likely to be simpler than the model of the whole, (b) we need not use the same technique of analysis for all constituents, and (c) elaborate and reliable models already exist for certain constituents.

(2) *A model of "macrorelations"* among these constituents so that the results of the isolated analyses can be combined to give an evaluation of the whole system behavior. This macromodel is likely to be a nontrivial one since the system has a "nonsummativity" character.

(3) *An estimation of the approximation* that such a decomposition may imply. Indeed, we are willing to make do with an approximate model if this is the price we must pay to control complexity. All we are then entitled to demand is that the degree of approximation remains known and, of course, tolerable.

Near-Complete Decomposability

A possibility of fulfilling these requirements is offered by the concept of *near-complete decomposability* and by its associated technique of aggregation of variables. The purpose of this monograph is to investigate in depth this possibility.

It is in economic theory that aggregation of variables has been most explicitly used as a technique to study and evaluate the dynamics of systems of great size and complexity. This technique is based on the idea that in many large systems all variables can somehow be clustered into a small number of groups so that: (i) the interactions among the variables of each single group may be studied as if interactions among groups did not exist, and (ii) interactions among groups may be studied without reference to the interactions within groups. This idea is of course quite general, and has, at least indirectly, been productive in disciplines other than economics; it is, for example, at the root of Boltzmann's fundamental hypothesis in statistical mechanics, of the Russell–Saunders approximation for the classification of atomic energy levels in quantum mechanics, and of Norton's and Thévenin's theorems in electric circuit synthesis.

This aggregation of variables yields rigorously correct results under two different types of conditions. The first type requires that interactions between groups of variables be independent from interactions within groups; then, these interactions between groups can be exactly analyzed without regard to the interactions within groups; the block stochastic systems studied in Chapter II and the lumpable Markov chain considered in Chapter III belong to the class of systems which satisfy the first type of conditions. The second type of conditions requires that variables be functions of variables of the same group only, i.e. that interactions between groups of variables be null. The system in this case can be said to be completely decomposable[1]: it truly consists of

[1] It is worth making the terminology more precise at the outset: such a system may be represented by a *completely decomposable* matrix, i.e., a square matrix such that an identical permutation of rows and columns leaves a set of square submatrices on the principal diagonal and zeros everywhere else; a *decomposable* matrix, as opposed to completely decomposable, is a matrix with zeros everywhere below the principal submatrices but not necessarily also above. *Near-complete decomposability* and *near-decomposability* are defined by replacing the zeros in these definitions by small nonzero numbers.