Algorithms for Games

G.M. Adelson-Velsky
V.L. Arlazarov
M.V. Donskoy

Algorithms for Games

With 38 Illustrations

Springer-Verlag
New York Berlin Heidelberg
London Paris Tokyo

G.M. Adelson-Velsky, V.L. Arlazarov,
and M.V. Donskoy
Institute for Systems Studies
9, Prospekt 60-Let Oktyabria
117312 Moscow B-312, USSR

Translator
Arthur Brown
P.O. Box 326
Garrett Park, MD 20766, USA

Library of Congress Cataloging-in-Publication Data
Adelson-Velsky, G.M. (Georgiĭ Maksimovich)
Algorithms for games.
Bibliography: p.
Includes index.
1. Game theory. 2. Algorithms. I. Arlazarov,
V.L. (Vladimir L'vovich) II. Donskoy, M.V.
III. Title.
QA269.A35 1988 519.3 87-23549

Translation of the original Russian edition
Programmirovanie Igr. © 1978, Nauka, USSR.

ISBN-13: 978-1-4612-8355-3 **e-ISBN-13: 978-1-4612-3796-9**
DOI: 10.1007/978-1-4612-3796-9

Text prepared by the translator using the Spellbinder® word processing software
and the STI Math Program.
Typeset by Science Typographers, Inc., Medford, New York.

9 8 7 6 5 4 3 2 1

Contents

Preface

Technicians, economists, industrial managers, and many other specialists often need to find an element belonging to a finite set and having certain given properties (provided, of course, that such an element exists). In principle, this problem can be solved by searching the set, element by element. There is, however, a well-rooted belief that this possibility is purely theoretical because the work involved in the search is enormous. Moreover, any general method (the search method included) may turn out to be either good or bad, depending on the concrete circumstances and the method chosen for implementation.

Suppose, for instance, that we need to find a root of the equation $x^4 + x^3 - 1 = 0$ on the interval $(0, 1)$ to within an accuracy of 0.1. Then it is better to evaluate the polynomial on the left side of the equation for the values $x = 0, 0.1, \ldots, 1.0$ than to use the exact Ferrari method, in which we first reduce the equation to the form $y^4 + py^2 + qy + r = 0$ by the substitution $x = y - 1/4$ and then transform this equation to the soluble form

$$\left|\left(y^2 + \frac{p}{2} + \alpha\right)^2 - 2\alpha\left(y - \frac{a}{4\alpha}\right)^2\right| = 0$$

where α is a root of the cubic equation

$$q^2 - 8\alpha\left(\alpha p + \alpha^2 + \frac{p^2}{4} - r\right) = 0,$$

after which we bring it into the form $\beta^3 + p'\beta + q' = 0$ by the substitution $\alpha = \beta - p/3$. Finally we use Cardan's formula

$$\beta = \sqrt[3]{\frac{-q'}{2} + \sqrt{\frac{q'^2}{4} + \frac{p'^3}{27}}} + \sqrt[3]{\frac{-q'}{2} + \sqrt{\frac{q'^2}{4} - \frac{p'^3}{27}}}\,.$$

In essence, a search amounts to the solution of problems arising from a given one when the value of an unknown parameter is fixed in one way or another, and a choice is made among a set of contemplated values that yields the most suitable solution. Often each of the contemplated problems, with the parameter fixed, is solved by a search. Then we speak of a multi-level or hierarchic search. If we adopt the most general definition of a search we can give only trivial recommendations for its implementation (though even these may be useful). Fortunately, the kinds of problems we are dealing with often have features in common aside from their solubility by searches. This permits us to establish, study, and apply general search methods.

This book aims to provide a concrete example of the programming of a two-person game with complete information, and to demonstrate some of the methods of solution; to show the reader that it is profitable not to fear a search, but rather to undertake it in a rational fashion, make a proper estimate of the dimensions of the 'catastrophe', and use all suitable means to keep it down to a reasonable size. The game programming problem would seem to be ideally suited to the study of the search problem, and in general for multi-step solution processes. The clarity and relative simplicity of the rules and the scoring of the results, the availability of suitable experimental methods for comparing various solution algorithms (including experiments on human approaches that arrive at answers produced by informal methods), all act to yield suitable methods for developing and trying out different approaches to the solution of problems by search methods.

A hierarchic search underlies all natural methods for finding a move in a game position. 'If I do this, I reach a position where various possibilities are open to me, but my opponent can answer thus and thus...'—this is a typical basis for the choice of a move. Clearly, it involves a search. As we shall show in the first chapter, however, a so-called *full-width* or *exhaustive* search need not require a complete enumeration of all possible positions that might arise by application of the rules of the game. We have in fact dedicated this book to the study of methods for limiting the extent of a search.

We have written about the programming of games (i.e. we have assumed that the reader would like to write a program that would play a game, serious or not) because we believe this is the best way to bring out the ideas and methods we are attempting to expound. We do not, however, pay attention to the technical problems of programming, essential though these may be. There is no specific technique for programming computers to play games. The fact that many technical paradigms were first conceived in the development of game programs does not contradict this assertion. These paradigms have later found wide development and application in the construction of many programs that are used in many different areas. Their prevalence attests to the fact that many clever programmers have been at work on them, and to the fact that they have stretched the capabilities of their machines to the limit.

We shall study games between two opponents named White and Black. Without loss of generality we may assume that these are zero-sum games. They permit at each stage a finite number of moves, for each of which the permissible replies of the opponent are known. Once the opponent's decision is made, a new position arises and is uniquely defined. Some positions are terminal: no decision is allowed to one of the opponents, the other wins and the one to move loses. If a non-terminal position arises, however, the opponent who has the move must in fact move.

We shall study a narrower class of games–namely games with complete information. In every such game we know which opponent is to move at each turn, and he must move. Thus both opponents know what the resulting position is. The positions in a game with complete information fall into three types: White to move, Black to move, and terminal positions. [We might define the notion of a game with several players and complete information, and carry over or suitably transform some of the results obtained in the book, but we shall not take the time to do so.]

Every game begins with a position which from now on we call the base position. In some games, for example card games, the base positions will vary, perhaps depending on some chance event. In other games, as for example chess, the base position is fixed once and for all. Even in chess, the player's concern is not with this fixed base position but rather with positions that arise either in the course of play or, as in problems and endgame studies, by artifice.

If the base position is prescribed, we may construct a game tree. Its nodes correspond to positions, and every arc leads from one position to another that can arise from it by a legal move. The base position corresponds to the root of the tree. A two-person game with complete information can be formally defined by prescribing the game tree, i.e. specifying the color of non-terminal nodes (designating the person who has the move in the given position) and the score corresponding to the just concluded move. These definitions are given in the first chapter and are widely applied throughout the book. To study the concrete properties of a game, however, we shall deal with positions and moves rather than with the game tree.

In Chapter 1 we develop the branch-and-bound method (the α,β-heuristic). We pay primary attention to the theoretical foundation of the method and to an estimate of the minimum size of the search required for its solution. In some places we relinquish an elegant inductive proof in favor of a more unwieldy one, in an effort to isolate the conditions needed for the existence of an objective score for the base position and for finding the best strategy for each of the opponents.

Chapter 2 is devoted to heuristic (i.e. inexact) methods for choosing a move in a contemplated position. We pay special attention to the problems of establishing such methods, and to a discussion of those properties of specific games that ensure good results when the methods are applied. It is worth noting that we employ a probabilistic approach in establishing our heuristic methods for programming games.

In Chapter 3 we develop the theory of analogical reasoning as a basis for decision-making. This theory is founded on the concept of a move that is independent of the position in which it is made; that is, the same move can be made in many different positions. The sequels to such a move may vary, but in many cases the move leads to roughly identical changes. Suppose that in the position B move Ψ is being studied. We want to know what conditions are sufficient to yield the same score for this move in another position C. We formulate these conditions and prove their sufficiency. They amount to this: the sequence of moves leading from B to C must have no influence on the variation that establishes the score for the move under study. In other words, the sequence must consist of moves that have no relationship to the decisive variation.

In Chapter 4 we take up the probabilistic approach to game programming. This approach has four aspects: a) the methods for formulating the elementary stochastic hypotheses and calculating the probability of correctly scoring a given position and finding the best moves; b) the methods for statistical testing of our hypotheses; c) the construction of more effective methods for computing the score and finding the best moves in a given position, on the basis of an analysis of a stochastic model of the game; and d) the probabilistic approach to the programming of games with complete information.

Since the probabilistic approach to game programming has only recently been applied, many of the results obtained in Chapter 4 must be regarded as preliminary, and some of them have not even been established—e.g. the statistical testing of the stochastic hypotheses and the basis of the probabilistic approach. Nevertheless, the results that have been obtained do in our opinion support the prospects for this new direction in game programming and we have therefore included the chapter.

At the end of the book we present an appendix containing a brief sketch of the work that has been done on algorithms for games and a bibliography which includes some references not cited in the text.

Chapters 3 and 4, which contain new results on two-person zero-sum games with complete information, are mutually independent. Formally they are also independent of Chapter 2, but we nevertheless recommend that the reader interested in probabilistic models of games should read Sections 1 and 2 of Chapter 2.

Since the problems we discuss have an immediate connection with programming, we have felt it worthwhile to introduce some notation taken from programming languages, in particular ALGOL. We use the assignment symbol ': =' and we denote the end of the definition of an operator by the semicolon ';'. Whenever this convention is in conflict with the ordinary rules of syntax we give precedence to the formal language.

We have devoted many years, and still devote our time, to the development of programs for playing chess. Quite naturally, this devotion has influenced our exposition; in particular, the examples we present are often related to chess. Nevertheless, with a few exceptions we have avoided the discussion of problems that arise uniquely in chess. Moreover, we are firmly convinced that one can write strong chess-playing programs without being a good chess player. Therefore, even when the examples presented deal purely with chess, the reader needs only an elementary knowledge of the game.

Translator's Note

The reader should be aware that the Russian text of this book was published in 1978, and that in the years since then, progress in the development of computer chess programs has been rapid. The book is still of interest since it deals with the ideas basic to the problems of search rather than with specific programs and programming techniques. Readers interested in recent developments should consult the publications of the International Computer Chess Association, (*Journal of the ICCA*, edited by H.J. van den Herik, Department of Informatics, University of Technology, Julianalaan 132, 2628 BL, Delft, The Netherlands). Other publications of interest include a recent paper by T. A. Marsland which cites the more important work in this field (*Computer Chess Methods*, *Encyclopedia of Artificial Intelligence*, Wiley, 1987, pp. 159–171), and Hartmut Tanke's very complete *Computer Chess Bibliography* (recently reissued in German). This bibliography can be obtained by writing to: Hartmut Tanke, Kienitzer Str. 104–106, D–1000 Berlin 44, West Germany.