Assessing data retrieval capabilities: a case study

C. B. Spengler and J. B. Westwood
Manchester Business School, Booth Street West, Manchester M15 6PB

A problem of data retrieval in a typical service company is described. A linear programming model is applied as an approximation and produces acceptable results. A simulation study is used for validation purposes and to identify the conditions under which the approximation is valid. The model's use as a planning device is outlined and the subsequent recommendations discussed.

(Received February 1977; Revised October 1977)

1. Background

One of the most common uses of the large computer is the handling of vast amounts of data and the process of retrieval can be quite complex, both in terms of search procedure and communication with the decision maker. There are many types of peripheral input/output devices (teletypes, visual display units, etc.) which have different operating characteristics and require different types of linkage with the processing unit. As the whole storage system expands, the limiting factor on efficiency may not be the actual amount of data stored, but the ability to complete the input, search, and output in the time available.

If the system is designed and constructed specifically for a particular job, then the hardware elements should be operating around their most efficient level. If, however, the system is built up in a less systematic way, there may be a mismatch of throughput capacity and the performance of the system as a whole will be limited by the slowest or least efficient element. The most critical questions to be answered are:

(*a*) Can a proposed system handle the envisaged demand?

(*b*) Which part of the system will become critical if demand rises?

(*c*) Which part of the system will be wasted if demand falls?

The information available to the decision maker consists of operating specifications for the hardware and some estimate of the demand profile.

The company in which this investigation was carried out and which exhibited the problems described above is involved in the supply of computerised information on insurance rates. A broker with an appropriate connection to the company's computer system 'asks' for information on insurance cover satisfying certain requirements. The data on store are searched and a list of the better policies within the given restraints is produced for the broker. At the time of the study, the company dealt only with life assurance and had 200 subscribers to this service. It was hoped to extend operations into the field of motor insurance, in which case it was estimated that a further 500 brokers would have to be catered for.

2. Problem definition

The suggestion from the management was that a second front end processor would enable the extra demand to be accommodated. They were naturally anxious to know the capabilities of any planned network and also whether investment in further hardware was justified. The envisaged system is shown in **Fig. 1.** The Interdata 15 was in operation at the time, and a Voicepac 2000 was available at a reduced cost if the offer was accepted within a fairly short period of time.

In theory there are 16 different routes through this network ($1 \times 2 \times 4 \times 2$) but several of these are infeasible. The relevant variables were identified and are described in **Table 1**; they are in fact the number of queries taking the particular route in a period of one hour.

The following constraints were also identified, the names being given in parentheses.

1. *Mainframe* (*IBMCPU*)

Let us assume that the time required for the search procedure will only vary with the type of information being retrieved. This was found to be 3 seconds for life and estimated to be 33 seconds for motor. This can be written in terms of the defined variables as:

$$3(\text{LIFID10} + \text{LIFID30} + \text{LIFVP10} + \text{LIFVP30}) + 33(\text{MTRID10} + \text{MTRID30} + \text{MTRVP10} + \text{MTRVP30} + \text{MTRVPSLO} + \text{MTRVPFST}) \leqslant 3{,}600 \quad (1)$$

2. *Front end processors*

(*a*) Interdata (IDCPU)—the time required for this machine to process a single character is 20 microseconds, so the hourly capacity is $1{\cdot}8 \times 10^8$ characters. Each printed line contains an average 60 characters and the number of lines per query is 10 for life and an estimated 15 for motor. This constraint can be written as:

$$600(\text{LIFID10} + \text{LIFID30}) + 900(\text{MTRID10} + \text{MTRID30}) \leqslant 1{\cdot}8 \times 10^8 \quad (2)$$

(*b*) Voicepac (VPCPU)—this can handle 176,000 characters per second giving an hourly limit of $6{\cdot}3 \times 10^8$ characters. The printed output requirements are the same as for the Interdata, but those for the audio mode are a little more complex. For each query, 240 characters are required if the operator is experienced; 480 if not. The constraint is, therefore:

$$600(\text{LIFVP10} + \text{LIFVP30}) + 900(\text{MTRVP10} + \text{MTRVP30}) + 480\,.\,\text{MTRVPSLO} + 240\,.\,\text{MTRVPFST} \leqslant 6{\cdot}3 \times 10^8 \quad (3)$$

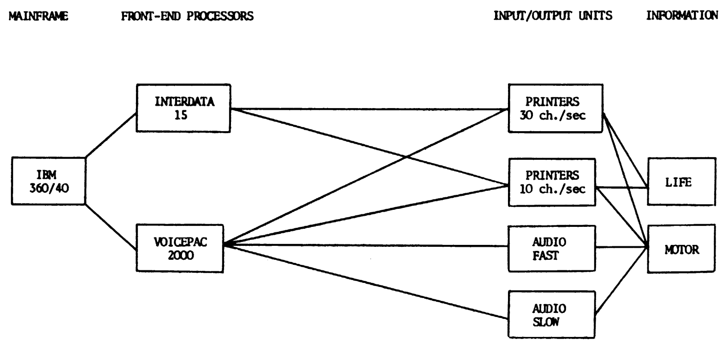

Fig. 1 Hardware of the proposed system

Table 1 Variables used in analysis

Variable name	*Description*
LIFID10	Life enquiry through 10 ch./sec input via Interdata
LIFID30	Life enquiry through 30 ch./sec input via Interdata
LIFVP10	Life enquiry through 10 ch./sec input via Voicepac
LIFVP30	Life enquiry through 30 ch./sec input via Voicepac
MTRID10	Motor enquiry through 10 ch./sec input via Interdata
MTRID30	Motor enquiry through 30 ch./sec input via Interdata
MTRVP10	Motor enquiry through 10 ch./sec input via Voicepac
MTRVP30	Motor enquiry through 30 ch./sec input via Voicepac
MTRVPSLO	Motor enquiry through inexperienced operator via Voicepac
MTRVPFST	Motor enquiry through experienced operator via Voicepac

3. *Input output devices*

Each query consists of three stages:

(*a*) enter data and verify

(*b*) search

(*c*) output results

and the input/output device is connected during the whole of the query. Empirical work suggests that if lines are operated above a certain loading level, delays experienced while attempting to gain access tend to drive away custom. This maximum tolerable loading factor was taken to be 67% of the theoretical throughput.

a) Interdata-print (IDLINES)—there are 34 lines from the Interdata, and so the total time available will be 3,600 × 34 × 0·67 seconds. The average query times are shown in **Table 2** and the constraint can be written as:

$$78(\text{LIFID10} + \text{MTRID30}) + 38 \,.\, \text{LIFID30} + 138 \,.\, \text{MTRID10} \leqslant 82{,}008 \quad (4)$$

(*b*) Voicepac-print (VPLINES)—for this mode, 18 lines are available and the average query times are those quoted in Table 2. The constraint is therefore:

$$78(\text{LIFVP10} + \text{MTRVP30}) + 38 \,.\, \text{LIFVP30} + 138 \,.\, \text{MTRVP10} \leqslant 43{,}416 \quad (5)$$

(*c*) Voicepac-audio (VPLINAUD)—there are nine lines available and the query times for experienced and inexperienced operators were estimated to be 83 and 153 seconds respectively. The constraint is:

$$153 \,.\, \text{MTRVPSLO} + 83 \,.\, \text{MTRVPFST} \leqslant 21{,}708 \quad (6)$$

4. *Customer constraints*

For each broker category, there is a minimum acceptable query rate. This is the number of queries per hour which the system must be capable of handling and is based on the assumption that during the busiest part of the day, each broker will make on average one query per hour. The categories are shown in **Table 3** together with the constraint names. The following equations can be written to represent these.

$$\text{LIFID10} + \text{LIFVP10} \geqslant 100 \quad (7)$$
$$\text{LIFID30} + \text{LIFVP30} \geqslant 100 \quad (8)$$
$$\text{MTRID10} + \text{MTRVP10} \geqslant 100 \quad (9)$$
$$\text{MTRID30} + \text{MTRVP30} \geqslant 150 \quad (10)$$
$$\text{MTRVPSLO} \geqslant 75 \quad (11)$$
$$\text{MTRVPFST} \geqslant 175 \quad (12)$$

The objective can be stated simply as the 'maximisation of the number of queries per hour which can be handled by the system, subject to equipment constrains and minimum broker requirements.' This can be written as:

$$\text{Max Z} = \text{LIFID10} + \text{LIFID30} + \ldots + \text{MTRVPSLO} + \text{MTRVPFST} \quad (13)$$

subject to equations 1-12 being satisfied.

3. Initial results

The two approaches to this type of network problem have utilised multilevel queuing models (e.g. Moore, 1975 and Pack, 1975) or probabilistic balance equations (see Gordon and Newell, 1967 and Langefors, 1970). One of the most critical factors which both types of model incorporate is the stochastic nature of the process. The arrival pattern of queries, especially when more than one type of information is being sought, can be very complex and this has meant that analysis of data flow can be extremely complicated and expensive. It was decided in this case to utilise the obviously linear characteristics of the problem and to treat the incoming calls not as stochastic variables but as decision variables. This assumption is obviously restrictive and requires testing.

To solve the problem as stated in the previous section, a standard linear programming package based on the revised simplex routine was used. No feasible solution was found to the problem as stated, so to test to what extent the proposed

Table 2 Query times for printed output

Query type	*Time required (secs)*
Life with 10 ch./sec teletype	78
Life with 30 ch./sec teletype	38
Motor with 10 ch./sec teletype	138
Motor with 30 ch./sec teletype	78

Table 3 Broker constraints

	Broker category	*Constraint name*	*Lower limit*
1	Life, 10 ch./sec teletype	LIFHARDSLO	100
2	Life, 30 ch./sec teletype	LIFHARDFST	100
3	Motor, 10 ch./sec teletype	MTRHARDSLO	100
4	Motor, 30 ch./sec teletype	MTRHARDFST	150
5	Motor, audio with inexperienced operator	MTRAUDSLO	75
6	Motor, audio with experienced operator	MTRAUDFST	175

Table 4 Output from the linear model

Variable	1 *Planned system*	2 *Motor rewrite*	3 *Faster mainframe*	4 *Motor rewrite and faster mainframe*	5 *Interdata failure*	6 *Voicepac failure*
LIFID10	100	100	100	90	0	100
LIFID30	0	0	0	0	0	1,537
LIFVP10	0	0	0	10	100	0
LIFVP30	100	100	100	967	480	0
MTRID10	91	100	100	100	0	100
MTRID30	0	25	25	0	0	150
MTRVP10	0	0	0	0	100	0
MTRVP30	0	0	0	150	150	0
MTRVPSLO	0	75	75	75	75	0
MTRVPST	0	175	0	175	230	0
Number of customers	291	575	400	1,567	1,135	1,887
Broker constraints violated	MTRHARDSLO MTRHARDFST MTRAUDSLO MTRAUDFST	MTRHARDFST	MTRHARDFST MTRAUDFST	—	—	MTRAUDSLO MTRAUDFST
Tight hardware constraints	IBMCPU	IBMCPU	IBMCPU	VPLINES IBMCPU	VPLINES VPLINAUD	IDLINES
% utilisation of IBM	100	100	100	100	86	96
Number of IDLINES	9	9	9	9	—	34
Number of VP print lines	2	2	2	18	18	—
Number of VP audio lines	0	9	5	9	9	—

system (Fig. 1) could handle the required load, the broker constraints were rewritten as 'less than' rather than 'greater than'. In this way a feasible solution will always be generated and this is shown in column 1 of **Table 4**. Four of the broker constraints are violated and only 291 queries are accepted in contrast to the required minimum of 700. The critical hardware constraint is seen to be IPMCPU and some way had to be found of increasing the mainframe speed or improving the search procedure. It would not be difficult to acquire a machine with twice the speed of the existing mainframe especially for this type of work, and the effect of this modification is shown in Table 4 column 2. The number of processed queries has risen to 400 and the system is again constrained by IBMCPU.

It was suggested at this stage that the search procedure for motor insurance could be rewritten to reduce the time from 33 to 8 seconds. This would take about six man months of labour but could be completed before the new system was to go online. The effect of this rewrite using the original mainframe is shown in column 3. The capacity has increased to 575 queries and only one broker constraint is violated, the mainframe being the limiting factor.

It seemed that a combination of motor rewrite and faster mainframe would easily handle the minimum broker requirements so the original 'greater than' constraints were reintroduced. The results (column 4) show that the system is easily capable of handling the peak demand and up to 1,567 queries could be handled provided 967 were of the life category and came from brokers with 30 cps terminals. The joint limiting factors are the mainframe (IBMCPU) and the number of lines through the Voicepac to printed output (VPLINES).

As the linear programming technique was being used in an unusual non-optimising role, it was decided to validate the model before proceeding with further analysis of the system.

4. Model validation

The data retrieval system was investigated using an event based simulation with the following characteristics:

(*a*) inter-arrival times were assumed to have a negative exponential distribution with mean appropriate for the various broker categories

(*b*) queries were treated on a first-come-first-served discipline

(*c*) queries not connected by the end of the simulation period were considered as lost.

Runs were carried out on the original planned system, the motor rewrite, and the combined motor rewrite/faster mainframe. These are shown in **Table 5**; the figures are averaged over one hour so that direct comparison with the linear programming results (in parentheses) is possible.

For the original planned system, there is little agreement between the two sets of results. Significantly fewer customers are serviced according to the simulation and this reflects the fact that customers are not decision variables as assumed in the linear model. The time required to get into the system is dependent on the numbers of the various types of input device and hence depends on the broker category. The waiting time for the mainframe is fairly constant over all customers.

This is to be expected as it is simply a single queue, single server arrangement. The departure from this behaviour by the MTRAUDSLO and MTRAUDFST classes of customer indicates a weak interaction between the time required to enter the system and the resulting length of the mainframe queue.

The simulation results for the motor rewrite show much better agreement with the earlier figures. The total number of customers served is close (4% difference), but the distribution among the customer classes shows some variation particularly in MTRHRDFST. The average time waited to enter the system is again dependent on the customer class whereas the mainframe queue time is fairly constant at almost two and a half minutes.

The rewrite/new mainframe run on the surface shows poor agreement but this is due to the fact that no account is taken in the linear model of the rate of arrival, only that a certain number of customers must be satisfied in one hour. The value 1567 shows what could be achieved if the customer classes had a suitable arrival rate. The value 689, which corresponds to full customer service relates to the actual arrival rate. Apart from the LIFHRDFST class, the linear output and the simulation show excellent agreement and both indicate that such a modified system is capable of handling the predicated customer demand profile. The waiting times exhibit the expected pattern but the existence of zeros would suggest that not all the Voicepac and/or the Interdata lines are required. This is substantiated by the linear model which for the rewrite/new mainframe system gives the number of Voicepac print lines as 18 (the maximum) but the number of Interdata lines as 9 (out of a possible 34).

A linear model is therefore an acceptable approximation when all constraints are satisfied. If customer constraints are included as 'less than' and the technology of the system is such that the minimum required throughput cannot be handled, the results will be misleading. Similarly if they are included as 'greater than', allowance must be made for slack constraints consuming customers whose arrival rate does not justify it.

Other information, such as waiting times, utilisation of individual hardware elements, etc. can obviously be obtained from a simulation study, but the cost of this type of analysis can be prohibitive. The linear programming approximation is cheap to run and provides a test bed for quickly evaluating alternative hardware systems.

5. Further analysis

When the initial results were produced, the company's management expressed some doubts about the necessity of having both the Interdata and the Voicepac. One justification for building such redundancy into the system is that the consequences of a front end failure are minimised. Columns 5 and 6 (Table 4) relate to the mainframe/rewrite modifications where, respectively, the Interdata and Voicepac machines are assumed to have failed. In the case of Interdata failure, the total number of queries falls quite sharply although all broker constraints are satisfied, i.e. the system can still operate at the minimum required level. On the other hand, if the Voicepac fails, the audio lines are automatically lost so that two broker constraints are immediately violated. The time released on the mainframe, however, enables a greater throughput to be achieved. The limiting factor in both cases is the number of lines available from the processor still in operation.

6. Recommendations

The computer runs described in Table 4 were augmented by further analysis of possible systems and the following conclusions were drawn:

1. the company's present system (life only, IBM slow mainframe, and Interdata processor) was, without major modification, incapable of handling the envisaged demand if a motor insurance service were offered.
2. if the decision to move into the motor insurance business was taken, the following modifications would be necessary:
   (*a*) a new mainframe and a rewrite of the search procedure
   (*b*) purchase of the Voicepac to be run in parallel with the Interdata
   (*c*) when the Interdata needs replacing, purchase of a similar machine
3. if the company can influence the choice of equipment made by the brokers then the 30 cps terminal should be recommended. This would increase the overall utilisation of the system without violating any constraints.

7. Conclusions

The company did in fact move into the new market along the guidelines suggested in recommendation 2(*a*) and 2(*b*). They are now at the stage of requiring a new processor and are using the linear programming approach to evaluate alternatives to a further Interdata machine. This emphasises the point that with careful interpretation, a linear programming approximation is acceptable when the system is relatively unconstrained, but will give meaningless results when the hardware constraints are tighter than those relating to service level.

Appendix Mathematical formutation of the data retrieval model

Consider a general data retrieval system as shown in **Fig. 2.** Let us define the following variables and parameters:

$x_{ijk\lambda m}$—number of processed queries per hour of information type i, through input/output connection type j of mode k, via front end processor λ to mainframe type m.

a_{im} —number of seconds required to sort for data type i on mainframe type m.

b_i —number of characters per query for information type i

Table 5 Simulation results

			Planned system				*Motor rewrite*				*Motor rewrite faster mainframe*			
Customer class	Number arrived		Number served		*Waiting times 1	2	Number served		Waiting times 1	2	Number served		Waiting times 1	2
1 LIF HARD SLO	90	(100)	33	(100)	256	928	90	(100)	0	142	90	(100)	0	54
2 LIF HARD FST	117	(100)	54	(100)	456	940	117	(100)	0	150	117	(967)	0	60
3 MTR HARD SLO	94	(100)	38	(100)	298	921	94	(100)	0	141	94	(100)	0	58
4 MTR HARD FST	143	(150)	54	(0)	260	871	143	(25)	0	142	143	(150)	0	59
5 MTR AUD SLO	73	(75)	6	(0)	95	1,075	46	(75)	373	143	73	(75)	268	59
6 MTR AUD FST	172	(175)	33	(0)	197	756	107	(175)	283	130	172	(175)	183	55
Total	689	(700)	218	(291)			597	(575)			689	(1,567)		

*1. Waiting time in seconds to make successful connection for data to be entered.
2. Waiting time in seconds between completion of data entry and access to the mainframe for processing.

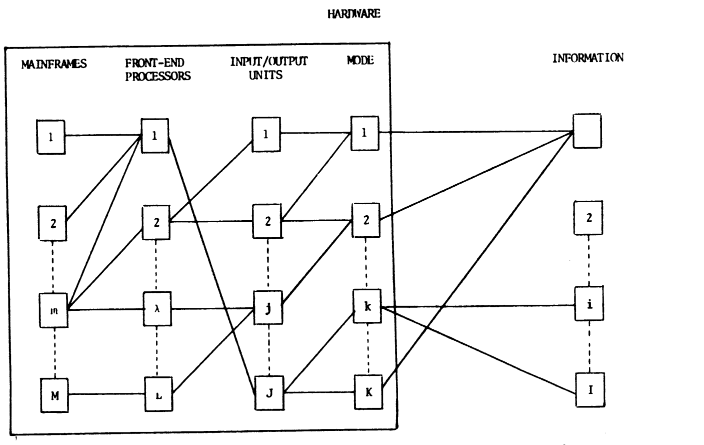

Fig. 2 General multi-unit system with sample connections

B_λ —maximum number of characters per second through processor type λ.

$c_{ijk\lambda m}$—average total time for a query of type i to be processed along the route j—k—λ—m through the system.

$n_{k\lambda}$ —the number of lines available into front end processor λ in mode k.

$D_{i\lambda}$ —the required minimum number of processed queries per hour of type i through input/output device j.

u —utilisation of lines.

The problem statement is then:

$$\text{Max } Z = \sum_{i,j,k,\lambda,m} x_{ijk\lambda m}$$

$$\begin{array}{lll}
\text{Subject to: } \sum\limits_{i,j,k,\lambda} a_{im}\, x_{ijk\lambda m} & \leqslant 3{,}600 & \text{all } m \\
\sum\limits_{i,j,k,m} b_i\, x_{ijk\lambda m} & \leqslant 3{,}600\ B_\lambda & \text{all } \lambda \\
\sum\limits_{i,j,m} c_{ijk\lambda m}\, x_{ijk\lambda m} & \leqslant 3{,}600.u.n_{k\lambda} & \text{all } k,\ \lambda \\
\sum\limits_{k,\lambda,m} x_{ijk\lambda m} & \leqslant D_{ij} & \text{all } i,\ i \\
x_{ijk\lambda m} & \geqslant 0 & \text{all } i, j, k, \lambda, m
\end{array}$$

References

GORDON, W. J. and NEWELL, G. F. (1967). Closed Queuing Systems with Exponential Servers, *Operational Research*, Vol. 15, pp. 254-265.

LANGEFORS, B. (1970). *Theoretical Analysis of Information Systems*, Student Litteratur, Lund, Sweden.

MOORE, S. C. (1975). Approximating the Behaviour of Non-Stationary Queues, *Operational Research*, Vol. 23, pp. 1011-1032.

PACK, S. D. (1975). The Output of an M/D/1 Queue, *Operational Research*, Vol. 23, pp. 750-760.

Book reviews

The Architecture of Concurrent Programs, by Per Brinch Hansen, 1977; 317 pages. (*Prentice Hall*, £13·55)

The programming language Concurrent Pascal is by now fairly well known, and has generated a lot of interest, as perhaps indicated by the official distribution of its implementation to 252 institutes. This book is concerned with the use of Concurrent Pascal in developing reliable concurrent programs and is divided into nine main sections. The first section discusses design principles, with Brinch Hansen presenting his requirements of the qualities a program should exhibit. Sections 2 and 4 provide a formal introduction of the main concepts and language notation of Concurrent Pascal (namely, processes, monitors and classes). Section 3 gives a short overview of Sequential Pascal, which is in the main identical to Wirth's Pascal, and can be used to program other system utilities (e.g. the compilers).

Sections 5 to 7 present three complete examples of working model operating systems written in Concurrent Pascal. As the text of these programs is included, these sections are rather long and form approximately half of the book.

Secion 8 provides the definition of Concurrent Pascal, together with the additional restrictions and extensions of the implementation that was distributed for the PDP11/45. As the compiler in this implementation generates code for a virtual machine, Section 9 describes the interpretive implementation of this virtual machine on the PDP11/45.

The book is written in the usual self congratulatory fashion of Brinch Hansen. The informed reader will have seen already several sections of this book published elsewhere. While the book claims to present a systematic way of developing reliable concurrent programs, this is illustrated solely through the use of Concurrent Pascal. It is clear that some of the methodologies advocated could be applied in other languages, for instance in the assembly languages that are still widely used (unfortunately). Indeed, the kernel which supports the Concurrent Pascal virtual machine has itself been programmed in assembler in the distributed implementation. This kernel has been extremely well engineered; however, Brinch Hansen only devotes 10 pages of the book to the discussion of its implementation. In contrast, the inclusion of three complete examples programmed in Concurrent Pascal seems somewhat excessive, and I feel that a reader would have benefited more from a more detailed description of what Brinch Hansen calls 'reliable machine programming'. However, the book is trying to sell Concurrent Pascal to replace assembly languages, and there perhaps a conflict arises: perhaps the book should have been titled *The Architecture of Concurrent Pascal Programs?*

P. A. LEE (Newcastle upon Tyne)

Computer Programming Methodology, by W. M. Turski, 1977; 208 pages. (*Heyden and Son*, £14·00)

Programming has reached the stage where there is a measure of agreement about the features of a 'good' program. Sometimes there is sound theoretical support for a concept, sometimes overwhelming evidence from experience, and sometimes nothing more than intuition. *Computer Programming Methodology* presents a thorough survey of current thought on both the design and construction of good programs. Apart from a preface and a page on 'Using this book', there are only four chapters, and the first of these is a four page introduction. The second and third chapters, each of over sixty pages, discuss basic operations and data structures, the way these basic constructs can be incorporated into modules and schemes for module interaction with due emphasis on parallelism and quasi-parallelism. The justification of the chosen constructs is theoretical where the theory exists and intuitive where it does not. Axiomatic methods are well illustrated in Chapter 2, which includes an overview of Guttag's work on axiomatic specification of data types.

The final chapter (72 pages) draws on the preceding material and tackles the problem of actually creating programs. For the individual programmer ways of achieving a predesigned structure are discussed, whilst for the project manager organisation of teams and documentation principles are dealt with too. Notes on reliability (as distinct from correctness), program improvement and maintenance complete the chapter, which, with its firm base in experience, complements the earlier more theoretical material.

In a book of this length there cannot be more than a brief mention of many topics, and the copious references are one of its best features. The book is nicely printed on good paper and my only major complaint is the index, which at 105 entries can at best be described as skimped.

A. P. BLACK (Oxford)