TR/12/88 December 1988

TOOLS FOR MODELLING SUPPORT AND CONSTRUCTION OF OPTIMIZATION APPLICATIONS

by

Gautam Mitra

TOOLS FOR MODELLING SUPPORT AND CONSTRUCTION OF OPTIMIZATION APPLICATIONS

GAUTAM MITRA

Department of Mathematics and Statistics, Brunel University,

Uxbridge, Middlesex, United Kingdom.

ABSTRACT

We argue the case for an open systems approach towards modelling and application support. We discuss how the 'usability' and 'skills' analysis naturally leads to a viable strategy for integrating application construction with modelling tools and optimizers. The role of the implementation environment is also seen to be critical in that it is retained as a building block within the resulting system.

1. INTRODUCTION

Computer based methods for supporting optimization applications are of great interest to operational research workers and management scientists. In this paper we put forward an analysis of the scope as well as the goal of such systems set against our understanding of the methodological, technological and organizational issues. We describe the new direction of research that we have embarked upon and provide an outline definition of the software tools that we have set out to develop.

A number of workers [Geoffrion 1988], [Bisschop 1988] have indicated the importance of the recent developments which take us beyond the considerations of robust optimization routines, and languages and systems for constructing mathematical programming models. The real life use of mathematical programming optimization models is one of many important examples of applying mathematical modelling. It is now well established that mathematical modelling in turn is but a particular instance of knowledge representation [Geoffrion 1985], [Mitra 1988]. In the fields of computer science and data processing there is a strong movement towards convergence of research directions. Thus methods of Al, database technology and programming language design are coming together [Brodie, Mylopoulos et al, 1984]. In the field of decision support systems these trends go even further [Mitra 1988] and management science and OR specialists are discovering close connections between their work and the research and developments in psychology, computer science and database technologies, as addressed to this topic. Although our own research objectives remain focussed on the well defined and also narrow topic of constructing applications using optimization techniques, we feel compelled to take into account substantial research results and software tools which are coming out of these fields [Mitra 1987]. Geoffrion [Geoffrion 1988] makes a strong case for modelling environments and lists a few desired characteristics which are (i) support of modelling life-cycle, (ii) equal access to policy makers and OR/MS analysts, (iii) a consistent vocabulary for model description, (iv) good management of key resources namely, data, models, and solvers within the system. Bisschop puts forward his

assessment of the issues with the following diagram (Diagram 1.1):

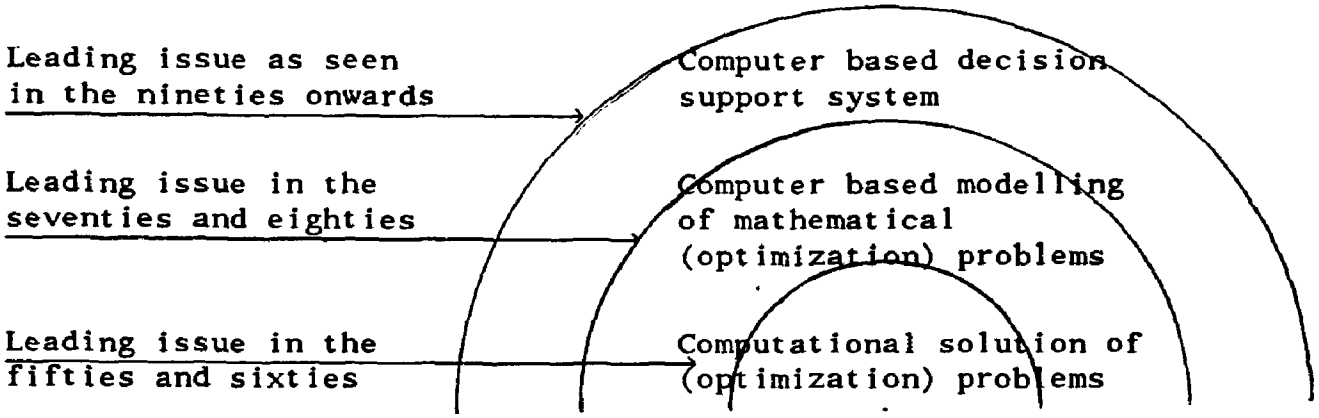

Diagram 1.1

This view is easy to explain and relate to. In his view the problems of optimization and modelling have been well addressed and many robust software tools for these can be found in the public domain. The development of an integrated Decision Support System is seen to be the leading research issue. We have also adopted a similar view of the software issues and put forward the argument that an 'open systems' approach should be adopted in the design of such systems. This approach also fits quite naturally with the layered view of the software items. For our purposes we define close and open systems in the following way. A close software system is one which has a well defined scope and applicability. It supports the user in his domain alone and it is difficult to extend its (re)use in other domains. In contrast, an open system is one which allows analysts, software engineer, end user, to make use of it for marginally different purposes and at different levels of competence. Scientific software libraries, graphics libraries, with well defined communication and control interfaces are basic building blocks of open systems [Iles and Hague 1988]. e

2. REQUIREMENT ANALYSIS : SCOPE, USABILITY AND SKILLS

In order to derive an outline specification of the system we consider the technology of the systems components, usability of the system and the skills level of the intended users.

o Technology: The system is designed to incorporate upto date devices and software components and requires high resolution bit-mapped screens as in advanced work stations, with a mouse or an alternative pointing device; it also suports integrated text 2-D graphics (colour) and pictures. It can also access data through networked distributed database and although voice interface and animation displays and 3-D graphics are not immediately included, the design features allow their incorporation in future,

o Usability and Skills Levels: Given that our objective is to define and implement an open system, we admit the very complex interaction between development and usage of the system. There are many constituents in these two roles and we categorize the constituents in four groups. We also introduce five different computer usage skills relevant to our analysis. In Table 2.1 we itemize these skills with a short code and in

Table 2.2 we set out the constituents together with their relevant skills and their job focus. Eason and Harker [Eason and Harker 1988] in their paper on user orientated approach to design, discuss these issues of skill and usability. The concept of end user computing in its own right admits many criticisms. Yet the ETHICS approach of Mumford [Mumford 1983] follows product development through analysis, specification, design and prototyping, delivery and use. This methodology of systems development through user participation, is now well established. Set against this background the role and relevance of our constituents and our case for an open system, may be fully appreciated.

Skills	Short Code
Supply data for decision problem. Interpret computer solution and implement decision within organization.	DECSIMPL
Provide rules, regulations and requirements and define domain model.	DOMNMODL
Construct a general mathematical optimization model.	GENERALM
High level programming and customization of application.	APPLPROG
System development and programming	SYSTPROG

Table 2.1 An Analysis of of Skills

Constituent	Skill	Job Responsibility
Problem Owner (End User)	DECSIMPL=Y DOMNMODL=N	Utilize the application (decision) support system and implement solution.
Domain Expert	DECSIMPL=? DOMNMODL=Y GENERALM=?	Work with the analyst/knowledge engineer to create a domain specific application.
Knowledge Engineer/ Analyst	DECSIMPL=N DOMNMODL=? GENERALM=Y APPLPROG=Y	Work with domain experts to create different applications
System Programmer	GENERALM=? APPLPROC=Y SYSTPROO=Y	Work with a variety of implementation vehicles to integrate application support and optimization tools.

Table 2.2 Indication of Skills: N = No, Y = Yes, ? = Questionable

o An Outline of the System and Its Use: In Diagram 2.1 we have illustrated how the layered software components make up the open system. There are four software layers which are optimizer/solver, modelling support system, application support system, and finally the application program.

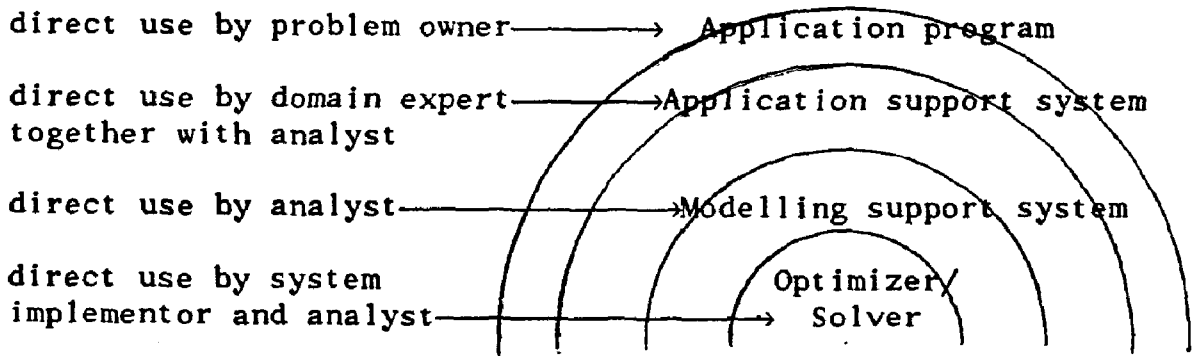

Diagram 2.1

The productivity and gearing achieved by such software tools is illustrated through Diagram 2.2. A discussion of the implementation environment which we wish to include as building blocks of the open system is postponed to section 5. The main players in the system are the analysts who use the modelling support system to create separate instances of models. The analyst, with a particular model instance then teams up with a domain expert to create an optimization application: crew scheduling, retail space planning are typical examples of these. Each application in turn serves a number of problem owners (end users).

In order to fix ideas and highlight the varying requirements we list a few well established applications of optimization models. These are set out in Table 2.3. Organizations which have made a commitment to the use of decision support systems usually have teams made up of analyst, domain experts and end users. Quite often a consultant or expert from the vendor company takes up the role of the analyst.

Application Domain	The Constituents	Description of Application
Bus Crew Scheduling	End User: Scheduling Team Domain Expert: Master Scheduler	Allocation of crews to bus time tables. Solution expected in extended bus timetable format with text and numbers.
Gasket Trim Minimization	End User: Shift Supervisors Domain Expert: Production Planner	Specify cutting of small rectangles out of large rolls or sheets to meet demand for parts. Solution required in report format with number, text and graphics.
Menu Formulation Problem	End User: Canteen Supervisor Domain Expert Diet Planner	A varied and planned menu to meet client demand. Solution expected with 2-D or 3-D graphics, text and possibly picture.
Shelf space allocation in retail sector.	End User: Department Heads Domain Expert: Shop Manager	Allocate floor space and shelf space to maximize selling of merchandize. Solution in text, graphics, 2-D, 3-D displays and possibly pictures.

Table 2.3 Representative Applications in Summary Form

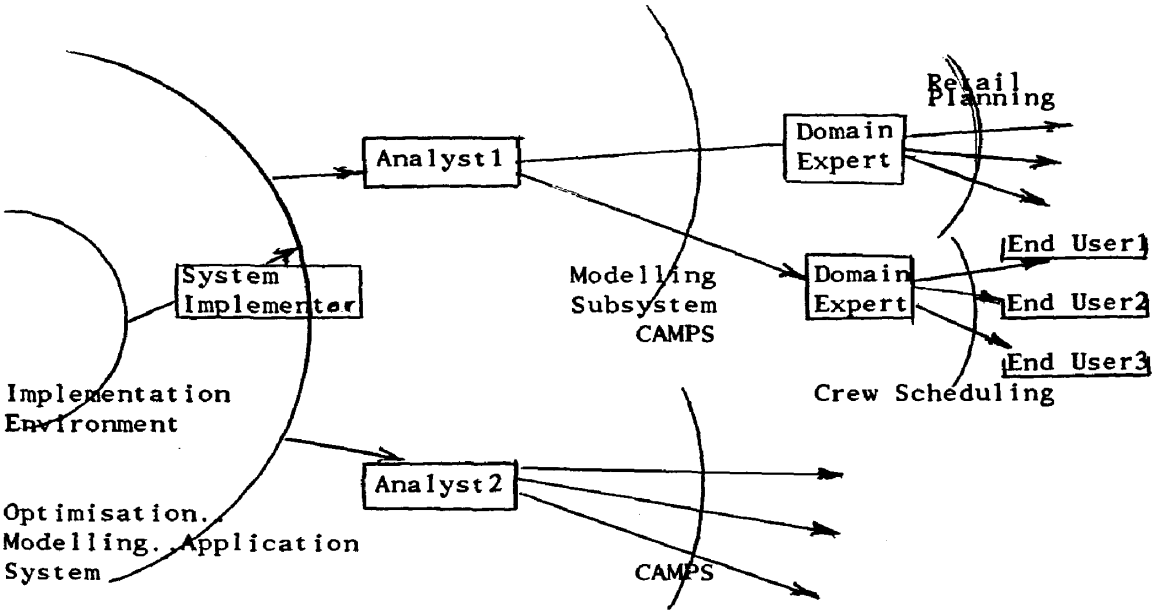

Diagram 2.2

3. MODELLING SUPPORT

Substantial development has taken place in the definition of mathematical programming modelling languages and a number of features have been established as of great value in the modelling process. In this section we present those key features which we consider are important in aiding the analyst in his task to construct mathematical optimization models.

o Model Description, Model Analysis and Solution Report: In most of these systems the models are described through a series of progressive and structured definitions of: Sets and basic entities, Data tables, Groups of decision variables, Groups of Constraints, Constraint relationships in linear form. In language based systems (UIMP, GAMS, AMPL) these are introduced using the language syntax and the keywords. Systems which make use of menus and screenforms, CAMPS, LPFORM, [Murphy et al 1986], the models are specified through interactive structured edit procedures. Most modelling systems also support simple reporting capabilities. The concept of model analysis, solution analysis, browsing and discourse are also pertinent at this level and has been well promoted by Greenberg [Greenberg 1983].

o Model Reformulation and Model Integration: Quite often it is simpler and more natural to describe a problem using logical variables and logical form for the relations. The methods of reformulating these into known MIP forms have been well discussed in [Darby-Dowman et al 1988], and [Williams et al 1988]. Many nonlinear programs can be also manipulated and reformulated into special ordered set type two form [Darby-Dowman et al 1988]. Murphy [Murphy et al 1986] makes a case for constructing and maintaining sub models such as production, inventory, transport and integrate these as and when appropriate. From an implementation viewpoint reformulation and model integration lead to the same issue of piecing together submodels.

Automating this task provides great support to the analyst.

o Model Validation: Model validation can take place at symbolic level and also at data level when data items are supplied. Bradley [Bradley and Clarence 1987] has highlighted the importance of introducing units of measurement in the definition of coefficients, model variables and constraints. Given that a set of complete unit conversion rules are also supplied unit validation of the restrictions can be automatically carried out. The coefficients themselves can be symbolically analysed for solvability or otherwise of the model [Brearley et al 1975]. Data items themselves are first checked against specified limits as determined by the application. Data items can also be used to establish solvability of the model as a follow up of the sumbolic analysis set out above.

o Model Documentation: Comparable with the requirement to document a computer program it is considered equally important to document a model. In a develpment environment it becomes necessary to communicate between analysts or between analyst and domain expert. Whereas in most language based systems the program with annotation is considered to be the documentation, in CAMPS we provide a separate utility to automatically document the model.

4. APPLICATION DEVELOPMENT

A customized system which supports a complete application with the model and the optimizer embedded in it, we call an application (decision) support system. We have identified application control, interface design (screens and menus), data and model management and discourse design, as essential aspects in the development of such a system. All these software features belong to the third layer of our system and they need to fit closely to the modeling layers.

o Application Control: This defines a complete set of end user commands by which the problem owner controls his application. These commands cover data entry and validation, error checking, model creation or revision, dispatch to the interactive or the batch queue to solve the model, and browsing of the solution returned by the solver,

o Interface Design: To start with we determine the nature of the interface and introduce text, graphics and other forms of communication. Specification of windows or screens, definition of structured edits or menu control by function keys, together with ‘HELP’ texts which are context specific, are the main tasks of interface design,

o Data and Model Management: Within an organization for corporate purposes data may be held or prepared in more than one department. The organization is likely to use different models driven by a subset of data items. The organization may simply use submodels which are appropriate to support the functions of a given department. The importance of a combined scheme for data and model management for these purposes is well discussed by Palmer et al of Exxon [Palmer 1984], Lucas of EDS [Lucas 1986] and Dolk [Dolk and Konynski 1985].

o Discourse Design: We see a convergence between the earlier generations concepts of diagnostic reports, exception reports, with the modern counterpart of diagnostic discourse, with full session summary and explanation procedure for decisions which are unclear to the problem owner. The discourse procedures are set out to convey through text, graphics, numbers and other communication vehicles many aspects of the model and solution to the end user. Greenberg has illustrated [Greenberg 1987] how end user discourse can be designed to explain LP models. We claim traditional solution reports often broken down into group requirements such as a financial report, production report and machine utilization report, can be similarly supported as sectoral views which Geoffrion calls Genus/module summary [Geoffrion 1987]. Since training an end user is genuinely a thorny issue we also plan to embed in the system a number of complete sessions of control and discourse as part of a training subsystem.

5. IMPLEMENTATION TOOLS

Our aim is to create an open system and for the first two software layers we have chosen standard (or at least well established) software items. We provide sufficient information concerning module definitions whereby different constituents can use these tools in their development tasks.

o Implementation Tools for Optimizer and Modelling Support: We have chosen FORTRAN, C and a low level screen package (CURSES) to implement FORTLP [Mitra and Tamiz 1988] and CAMPS respectively. Our choice is based mainly on efficiency and portability considerations. A complete statement of an LP model may be viewed as a declarative knowledge of the underlying physical problem. It is interesting to note that many combinatorial problems such as crew scheduling, vehicle routing and cutting stock problems, require activity generations (duties, routes, patterns, respectively) which can be only done by procedural methods. These examples also highlight the need to provide a programmer's interface to the generated code or at least the model generator statements of the modelling system. We are considering implementing such an interface. We would also like to highlight the work of Ladhelma and his colleagues [Ladhelma 1988] who have designed and implemented a mathematical modelling environment (MME) with a functional language MPL. They have created a very credible integrated modelling and optimization system.

o Tools for Application Generation: We have considered a number of Fourth Generation Languages and Knowledge Based Systems Shells as vehicles for implementing the outer layer of our application support software. We refer the readers to [Martland 1986], [Holloway 1988], and [Mettrey 1987] for over views of alternative 4GL and KBS products. 4GL and KBS both suffer from the drawback that there is a long learning time to make full use of these systems, execution, speed, and portability, are also problematical. On the other hand they provide powerful tools for display management and data management. The work of Markowitz [Markowitz et al 1984], EAS-E system

seems highly relevant in this context. Since 4GL's do not support knowledge representation facility we have not considered them any further. We are, however, committed to use a suitable database system.

We are evaluating a number of knowledge based systems shells and our arguments for adopting a KBS product (or products) are set out below. Logic programming, production rules, frames, are perhaps the most useful knowledge representation method. In the MIP reformulation task, use of such .an approach (PROLOG [Williams et al 1988]) has already been demonstrated. We also see very good use of production rules in validating data limits, consistency of units of measurement. CAMPS as a modelling tool uses the concepts of structured edit and program generation. The use of frames (structured objects with slots) at outer level can be used in a natural way to provide interface with the modelling system. Our target of dialogue support covers use of text, graphics, icons and even pictures. We expect to make use of frames and dialogue support tools of KBS shells. An example of application specific software architectures along these lines is already reported by Dempster [Dempster 1988] where he has used MINOS as the optimizer and KEE as the application as well as modelling of environment. Currently we are evaluating three such tools namely KEE (Knowledge Engineering Environment) [Fikes and Kehler 1985], ART (Automated Reasoning Tool) [Mettrey 1987], and LEONARDO [Jones and Graham 1988]. We also see the requirement for a separate PROLOG compiler and suitable interface definition as none of these tools support this facility. As structured objects and object oriented programming have emerged as an important method of knowledge representation, we are also considering whether or not SMALLTALK [Goldberg 1983] system can be introduced in this outer layer and used in the application development.

The two inner layers are concerned with mathematical description of the models and deriving computational solutions. The outer layer is concerned mainly with the domain expert and the end user. It has the primary requirement of capturing domain specific knowledge and supporting interaction and dialogue. As a result the possibility of using hypertext systems are also of interest to us. We find the facilities within KMS hypertext system [Akscyn et al 1988] particularly attractive. Within this system it is possible to combine structured edit, navigation through the system, and also execution of images (program modules) at nodes. This might provide us with a rich implementation environment within which our software layers, along with the tools, may coexist.

We wish to conclude this section by justifying why we are evaluating such a variety of implementation tools with different focus and capability. Our main argument is that as hardware and software technology progress it becomes impossible to access them (e.g. natural language communication, multimedia systems) without defining a suitable control and interface structure. The only way to define such an interface is to learn and

make use of the corresponding implementation vehicles. This observation also reinforces our case for an open system.

6. DISCUSSIONS

In this paper we have outlined the preliminary findings of our longer term research project of defining and constructing integrated tools which support optimization applications. We present requirement and usability analyses of the system and discuss our implementation strategy. There are many aspects of design and implementation that we are still exploring. Constructive criticisms on all aspects of this paper covering focus, definition and implementation strategy will be gratefully entertained and acknowledged in our future works.

ACKNOWLEDGMENTS

Our research in the field of modelling support system and optimization applications has been supported by the UK Science and Engineering Research Council, and US Army's European Research Office. Both these grants supported Dr C Lucas who implemented the CAMPS modelling system. NATO Scientific Affairs Division have supported our collaborative research with Dr H Greenberg. NAG Ltd have also maintained a strong interest in our work and are now working with us closely to define the next generation of software products. We have greatly benefitted from discussions with many colleagues. In particular we would like to mention H Greenberg, A Geoffrion, H P Williams, J Bisschop, P L K Jones, F Murphy, G Bradley, and M A H Dempster, whose innovative ideas have, in one way or another, been incorporated in our work.

REFERENCES

Akscyn, R M., McCracken, D L., and Yoder, E A., (1988). KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations, Comm ACM,Vol31,No7,pp820-8335.

Bisschop, J.J., (1988), A Functional Description of an Integrated Modelling Software for Mathematical Programming, (Aug), presented to the 13th Internaitonal Mathematical Programming Symposium, TOKYO.

Bradley, G H., and Clarence, R D., (1987). A Type Calculus for Executable Modelling Languages. IMA Journal of Mathematics in Management, Special Issue on Mathematical Programming Modelling Systems, Guest Editor: G Mitra, Vol 1, pp 277-291.

Brearley, A L., Mitra, G., and Williams, H P., (1975). Analysis of Mathematical Programming Models Prior to Applying the Simplex Algorithm. Math Prog., Vol 8, Pp,54-83.

Cunningham, K., (1986). Optimization Models with Spreadsheet Programs, Univ of Chicago, Technical, Report.

Darby-Dowman, K., Lucas, C, Mitra, G., and Yadegar, J., (1988). Linear, Integer, Separable and Fuzzy Programming Problems: A Unified Approach Towards Reformulation. Journal of the OR Society (GB), Vol 39, No 2, pp 161-171.

Dempster, M A H., et al (1988). Expert Financial Systems for Debt Management in [Mitra1988].

Dolk, D R., and Konynski, B., (1985) Model Management in Organizations, Information and Managemeent, Vol 9, No 1, pp 35-47.

Eason, K.D., and Harker, S., (1988), The Supplier's Role in the Design of Products for Organisations, The, Computer, Journal, (UK), vol. 31, No. 5, pp. 426-430.
Fikes, R., and Kehler, T., (1988). The Role of Frame Based Representation in Reasoning, Comm. ACM, Vol. 28, No. 9, pp. 904-920.
Fourer, R., Gay, D M., and Kernighan, B W., (1987). AMPL: A Mathematical Programming Language. Computing Science Technical Report No 133, AT & T Bell, Labs, Murray, Hill, NJ, USA.
Geoffrion, A M., (1985), Private communication, 12th International Mathematical Programming, Symposium, Boston.
Geoffrion, A M., (1987). An Introduction to Structured Modelling, Management Science, Vol. 33, No. 5, pp. 547-588.
Geoffrion, A M., (1988), Computer Based Modelling Environments: A Road to Greater Productivity, Quality and Popularity for Management Science /Operations Research, (May-Aug), Keynote speech to Canadian Operations Research Society meeting, Montreal.
Goldberg, A., and Robson, D., (1983). Smalltalk-80: The Language and Its Implementation. Addison. Wesley, Reading, Mass.
Greenberg, H J., (1983). A Functional Description of ANALYZE: A Computer-Assisted Analysis System for Linear Programming Models. ACM Transactions on Mathematical. Software, Vol. 9, pp. 18-56.
Greenberg, H J., (1987). A Natural Language Discourse Model to Explain Linear Programs, Decision, Support. System, Vol.33, pp. 333-342.
Greenberg, H J., Lucas, C, and Mitra, G., (1987). Computer Assisted Modelling and Analysis of Linear Programming Problems: Towards a Unified Framework, IMA Journal. of, Mathematics, in, Management, Vol, 1, pp, 251-265.
Holloway, S., (1988), Editor. Evaluation of Fourth Generation Systems, UNICOM Information, Technology, Report, Series, Kogan, Page, UK.
Iles, R., and Hague, S., (1988), Knowledge Based Front End Research Project: FOCUS, Internal, report, NAG, Ltd.
Jones, P L K., and Graham, I. Expert Systems: Knowledge, Uncertainty and Decision, Chapman, and, Hall, 1988.
Lahdelma, R., (1988). MME: A Mathematical Modelling Environment, presented to EURO IX, Paris. Report of Nokia Research Center, PO Box 780, 00101 Helsinki, Finland.
Lucas, J., (1986). Expert System/Mathematical Programming Applied to Strategic Decisions. Paper presented at TIMS XXVII, Gold Coast, Australia, and runner up Franz, Edelman, Award, TIMS.
Markowitz, H M., et al, (1984). The EAS-E Application Development System: Principles. and, Language Summary, Comm ACM Vol 27, No.8, pp. 785-799.
Martland, D., (1986), Editor. Fourth Generation Systems, UNICOM - Technical Press Report, Series, Gower. Press, UK.
Mettrey, W., (1987). An Assessment of Tools for Building Large Knowledge-Based Systems, AI-Magazine, winter. 1987, pp. 81-89.
Mitra, G., (1987), Mathematical Programming Modelling Systems, Guest Editor Mitra, Special Issue of IMA Journal of Mathematics in Management, vol 1, No 3 & No 4.
Mitra, G., (1988), Editor, Mathematical Models for Decision Support, Proceedings of NATO. Advanced, Study, Institute, Springer Verlag.
Mitra, G., and Tamiz, M., (1988). FORTLP: Linear and Integer Programming System, User, Reference, Manual, Brunel, University, and, NAG, Ltd.
Mumford, E., (1983), Designing Human Systems for New Technology; the ETHICS Method, Manchester, Business, School.
Murphy, F H., and Stohr, E A., (1986). An Intelligent System for Formulating Linear Programs. Decision. Support. System, Vol. 2, pp. 39-48.
Palmer, K., et al (1984). A Model Management Framework for Mathematical Programming, Wiley, New. York.
Williams, H P., and McKinnon, K I M., (1988). Constructing Integer Programming Models by Predicate Calculus, (Aug). Presented to the 13th International Mathematical Programming Symposium, TOKYO.