RAIRO. RECHERCHE OPÉRATIONNELLE

JAMES K. HO
ÉTIENNE LOUTE

Computational aspects of DYNAMICO : a model of trade and development in the world economy

RAIRO. Recherche opérationnelle, tome 18, n° 4 (1984), p. 403-414

<http://www.numdam.org/item?id=RO_1984__18_4_403_0>

R.A.I.R.O. Recherche opérationnelle/Operations Research
(vol. 18, n° 4, novembre 1984, p. 403 à 414)

COMPUTATIONAL ASPECTS OF DYNAMICO: A MODEL OF TRADE AND DEVELOPMENT IN THE WORLD ECONOMY (*)

by James K. Ho (1) and Étienne Loute (2)

Abstract. — *Project DYNAMICO has been developed at the United Nations Secretariat to study the interaction between trade and development in the world economy. It involves application of the Dantzig-Wolfe decomposition method to a multi-regional (block-angular) linear programming model. Costa and Jennergren* (1982) *presented the methodological features of the project, justification of the decomposition approach, sample results of preliminary runs, as well as indications of the complexity, difficulties and limits to the actual implementation of the system. In this paper, computational results of applying more sophisticated decomposition software to DYNAMICO are reported. It is shown that the requirement on computing resources can be reduced tremendously, to the extent that using even less than current facilities, the actual static model can be operated inexpensively, expanded to allow for more detail or extended to a truly dynamic optimization model. Such observations lead to suggestions for further development of DYNAMICO. They should also be useful for the design and implementation of any similar project in multiregional or multidivisional policy modeling.*

Keywords: Linear Programming; Decomposition; Economic Modeling.

Résumé. — *Le projet de modélisation économique DYNAMICO a été développé à l'ONU dans le but d'étudier l'interaction entre le commerce et le développement économique mondial. Il s'agit d'appliquer la méthode de décomposition de Dantzig et Wolfe à un modèle multirégional mis sous forme de programme linéaire ayant une matrice des contraintes à structure bloc-angulaire. Costa et Jennergren* (1982) *ont présenté les aspects méthodologiques du projet tout en justifiant l'utilisation de la méthode de décomposition; ils ont donné, de plus, des résultats préliminaires, des indications sur la complexité, les limites du système actuel ainsi que les difficultés auxquelles ils se sont heurtés. Dans cet article, nous décrivons quelques expériences numériques d'application de logiciels avancés de décomposition à DYNAMICO; nous montrons que l'on peut réduire très sensiblement les ressources informatiques mises en œuvre dans la mesure où, avec des ressources même inférieures à celles dont nous disposons actuellement, le coût de traitement du modèle statique est très modeste, ce qui laisse la possibilité d'introduire davantage de variables pour le rendre plus explicatif ou de le transformer en un modèle dynamique. Ces remarques seront utiles tant pour le développement ultérieur de DYNAMICO que pour des projets similaires de modélisation de politiques multirégionales ou décentralisées.*

Mots clés : Programmation linéaire; Décomposition; Modélisation Économique.

(*) Received June 1983.
(1) Management Science, College of Business Administration, The University of Tennessee, Knoxville, TN 37996, U.S.A.
(2) Facultés Universitaires Saint-Louis, Bruxelles, Belgium.

R.A.I.R.O. Recherche opérationnelle/Operations Research, 0399-0559/84/04 403 12/$ 3.20

1. INTRODUCTION

Large-scale linear programming (LP) models are very often highly structured. The best known example must be multiregional models where independent submodels are coupled through linking constraints that represent, e. g. interregional trading or resource sharing. The resulting coefficient matrix then has the block-angular structure. The Dantzig-Wolfe decomposition principle [4] applied to this class of problems leads to a finite procedure that seeks the global optimal solution by coordinating a sequence of solutions to the submodels. As submodels are processed one at a time, one expects to be able to solve larger models than allowed by a direct approach. Moreover the process of coordination has economic interpretation as price-directive, decentralized resource allocation.

Recently, Costa and Jennergren [3] reported on their preliminary experience with project DYNAMICO at the United Nations Secretariat which models the trade and development of the world economy as a ten-region linear program. In section 2 we summarize the characteristics of their model. From their concluding remarks, it is apparent that the actual implementation, which has been developed without the benefit of any advanced linear programming computational techniques turns out to be rather cumbersome and expensive to operate. Yet, their effort is truly significant in demonstrating decomposition as a modeling approach. It is this observation that motivated the present work which investigates how state-of-the-art decomposition software would perform on DYNAMICO and hence suggests directions for any future development. In section 3 we briefly describe recent research on advanced implementations of decomposition algorithms. See e. g. Ho and Loute [6, 7]. Specifically, two decomposition codes are introduced. The first one, called DECOMP, is in Fortran and easily portable. The second, called DECOMPSX is much more sophisticated and robust, but is software specific and machine dependent. Both codes use the same input format which is an obvious and logical extension of the by now conventional MPS format. Section 4 presents computational results of applying DECOMP and DECOMPSX to a typical case of DYNAMICO. It is observed that in its existing computing environment DYNAMICO can be solved very efficently using either DECOMP or DECOMPSX. Ramifications of this observation in future development of DYNAMICO are discussed in the concluding remarks in section 5. Although the experience reported in this paper is specific to the DYNAMICO model, it should be of general interest to any policy modeller contemplating decomposition as an approach to multiregional or multidivisional problems.

2. DYNAMICO: THE MODEL, THE SYSTEM, AND INITIAL EXPERIENCE

We give only a brief summary here to make the paper more self-contained. For a more detailed description, the reader is referred to Costa and Jennergren [3]. For a given year the model covers ten regions in the world economy: (1) North America, (2) Western Europe; (3) Soviet Union; (4) Eastern Europe; (5) Japan and Oceania; (6) Latin America; (7) Oil-exporting North Africa and Middle East; (8) South-Saharan Africa; (9) Market Economies of Asia; and (10) China and Centrally Planned Economies of Asia. In each region, ten sectors are considered: (1) agricultural products, excluding grains; (2) grains; (3) other primary products (raw materials), excluding energy; (4) energy products, excluding petroleum; (5) petroleum; (6) intermediate goods; (7) consumer goods; (8) investment goods (industrial manufacturing); (9) construction and (10) services. Each sector, except construction, produces a commodity that is tradable with other regions through import and export. The regional economies are governed by constraints on (1) material balance; (2) labor and land usage; (3) demand for capital; (4) supply of capital; (5) investment expenditure; (6) consumption expenditure; (7) macroeconomic definitions; and (8) policy for import, export, and trade balance.

The coupling (or global) constraints are simply market clearing conditions on the interregional trading of the nine tradable commodities. The objective function to be maximized is taken to be the gross world product (GWP) which is defined as the sum of regional GNPs.

Each regional submodel has between 60 to 70 constraints, of which about 20 can be expressed as simple bounds. The number of variables is about 80, of which only about 50 are so called structural variables, the others can either be expressed as slacks (logical variables) or are exogenous, with values to be fixed at historical levels.

To better illustrate the overall structure of the complete model, the following highly abbreviated mathematical formulation is used.

For each region r, let:

$$Y_r \equiv \text{GNP},$$

$$e_r \equiv (e_{r1}, \ldots, e_{r9}): \text{ export of tradable commodities;}$$

$$m_r \equiv (m_{r1}, \ldots, m_{r9}): \text{ import of tradable commodities;}$$

$X_r \equiv$ variables for:

(i) production of commodities by various technologies,

(ii) investment in capital,

(iii) demand for capital,

(iv) capital stock,

(v) private consumption,

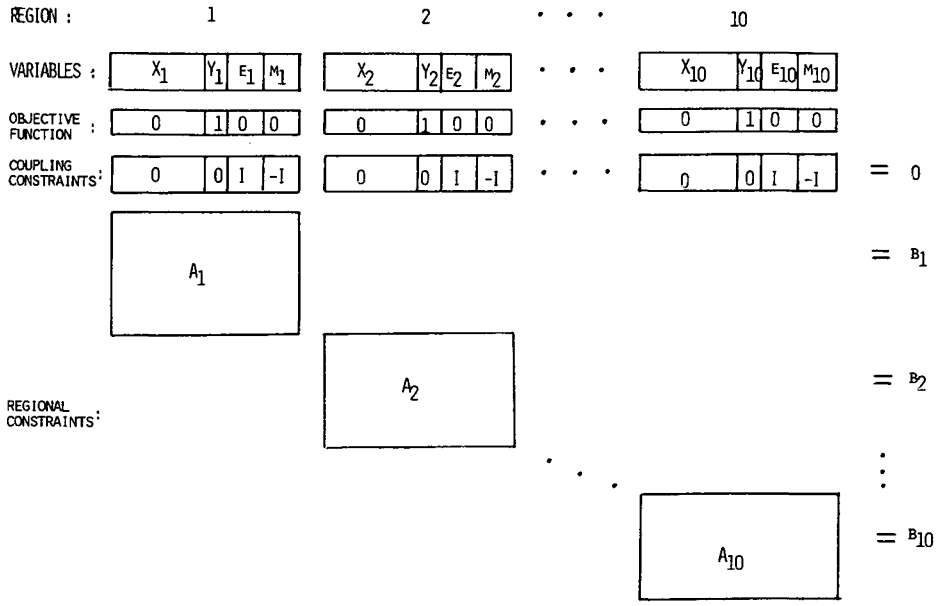
(vi) government consumption, and

(vii) trade balance;

$b_r \equiv$ logical values or historical data;

$A_r \equiv$ matrix for the eight types of regional constraints.

Then the objective function to be maximized is $\text{GWP} \equiv \sum_{r=1}^{R} Y_r$. The LP for DYNAMICO in detached coefficient form, which has the block-angular structure, is illustrated in the figure.



Structure of the LP for the DYNAMICO model.

A brief description of the *actual* system for DYNAMICO implemented at the United Nations is now given. The system consists of a Prelink and a Postlink phase (i. e. Phase 1 and Phase 2 in LP parlance). Using various objective functions, fifteen solutions from each region are generated in the Prelink phase. Therefore, a total of 150 proposals are used to initiate the master (global) problem in the Postlink phase. It should be remarked that DYNAMICO requires that this first master problem be feasible. With new prices on the tradable commodities from the master, each submodel is solved

for a new proposal, and so on. It is reported that typically, less than eight iterations are required to arrive at a near optimal solution, although no indication of the convergence tolerance used has been given.

The solution to the yearly model obtained above is then used to define historical values for the exogenous variables in the model for the following year. This way, an entire trade and development path for the world economy can be generated. (Note that in this sense, DYNAMICO is not truly a dynamic optimization model.)

DYNAMICO has been implemented on an IBM 370/168. It requires about 2,000 K bytes memory and although no actual statistics are reported, Costa and Jennergren [3] gave the impression that computational times are lengthy, interaction with the database is cumbersome, and operating the system can become frustrating and painstaking. Yet their experience is a successful demonstration of decomposition as a modeling approach. The main reason for DYNAMICO to be inefficient as a software package is that it has not incorporated any advanced techniques in linear programming, e. g. sparsity, efficient inversion, pricing and updating. In the next section, we review some recent development in research on the decomposition method.

3. ADVANCED IMPLEMENTATION OF DECOMPOSITION ALGORITHMS

3.1. Overview

The Dantzig-Wolfe decomposition principle leads to finite algorithms which involves the iterative coordination of solutions to the subproblems. Since the subproblems are usually much smaller than the complete one, the decomposition process may be more efficient than a direct approach like the simplex method. Therefore, since its introduction in 1960, decomposition has always been an appealing idea in terms of deriving efficient computational tools for large-scale linear programming. However, subsequent attempts in implementation and application have led to inconclusive, and often less than encouraging results. See Dirickx and Jennergren [5] for a summary. A frequent observation is that convergence, especially near optimality is very slow. That means it takes many iterations to achieve an optimal coordination of the subproblems. Cases where few iterations are required have been observed. But they are the exceptions rather than the rule. Nowadays, textbooks on linear programming mention decomposition more or less as an elegant idea that does not seem to perform well, at least not well enough computationally to warrant common practice. Meanwhile, variants of the simplex method have undergone tremendous advances mainly due to sparse matrix techniques and are widely imple-

mented in powerful commercial software packages. This makes the need to further develop decomposition technique less urgent and the task of arriving at meaningful comparisons more difficult.

Nevertheless, as long as there are problems for which decomposition is judged to be the appropriate *modeling* approach, and as long as there are problems for which decomposition works well as a computational tool, it should remain as a viable option in linear programming. And just as important, efforts to identify and characterize appropriate applications should be encouraged. Moreover, there is another aspect to this approach that has by and large been overlooked. This is the eventual mode of application. So far, most algorithms for linear programming have been developed as "black box" solvers, with little possibility of logical intervention by the user. By logical intervention, we mean applying whatever knowledge (either *a priori*, but not already incorporated in the formulation; or interactive, as learned in process) the user has about the problem to help improve performance of the algorithm. Little can be done with the direct simplex approach, except perhaps in variation of the pricing strategy. In many applications, especially policy modeling, the user may know, for instance, that certain submodels will be more active than others. It would therefore be helpful to be able to direct the algorithm to concentrate more on the active components. In other cases where parameters have to be "fine tuned" before the model is finalized, the direct accessibility of submodels may be most useful. Note that such fine tuning may also help to reduce the number of iterations. Such flexibility is possible by decomposition.

3.2. Specifications

Having established that the need to solve linear programs by decomposition exists, the question is naturally how to do it as efficiently as possible. The answer, at least in principle, is quite simple. Since the decomposition algorithm involves solving both the master and subproblems as linear programs, one should apply the most advanced LP software available to these. The rest is proper data management: setting up the LP to be solved, transferring and updating data according to the information flow dictated by the algorithm. Here, we distinguish between two different levels of advanced LP software. One is the Fortran level, which is a reasonable assumption for most academic research projects. An advanced LP code in Fortran would be one with sparse matrix storage and a basis inversion routine to maintain a sparse representation of the basis inverse. See, e. g. [2] and the references therein. The second level is that of commercial software. Typically, this includes in addition to the features mentioned above, sophisticated data structure, control

of numerical stability, complex pricing and pivot selection schemes, all professionally designed, implemented and documented. See, e. g. [1]. In the following we shall introduce two implementations of the decomposition method compatible with the respective level of advanced LP software.

First, the algorithm, as well as several computational strategies found to be indispensible for an efficient implementation will be summarized. A block-angular LP with R blocks has the form:

$$\text{LP} \quad \begin{cases} \text{maximize } z = \sum_{r=1}^{R} c_r x_r, & (1) \\ \text{subject to } \sum_{r=1}^{R} B_r x_r = d_0, & (2) \\ A_r x_r = d_r, \quad r = 1, \ldots, R, & (3) \\ x_r \geqq 0, & (4) \end{cases}$$

where c_r is $1 \times n_r$, d_r is $m_r \times 1$ and all other vectors and matrices are of suitable dimensions.

Assuming that each block of constraints (3) and (4) is feasible, the m_0 constraints in (2), called the coupling constraints, will be used to define a master problem which, at cycle k of the algorithm, determines prices π_0^k on these constraints. With these prices, a subproblem from block r at cycle k (SP_r^k) can be defined:

$$\text{SP}_r^k \quad \begin{cases} \text{maximize } v_r^k \equiv (c_r - \pi_0^k B_r) x_r, & (5) \\ \text{subject to } A_r x_r = d_r, & (6) \\ x_r \geqq 0. & (7) \end{cases}$$

If SP_r^k is bounded, then it has an extreme point optimal solution. Otherwise, a solution corresponding to an extreme ray (henceforth called simply an extreme ray solution) is obtained. In either case, denote the solution by x_r^k, and if it passes a test of candicacy, a proposal for the master problem is generated. The proposal from SP_r^k, defined as:

$$\begin{bmatrix} p_{rk_r} \\ q_{rk_r} \end{bmatrix} \equiv \begin{bmatrix} c_r x_r^k \\ B_r x_r^k \end{bmatrix}, \tag{8}$$

is the (k_r)-th proposal submitted by block r through cycle k. Representing all the proposals from block r as a matrix, we have:

$$\begin{bmatrix} P_r^{k_r} \\ Q_r^{k_r} \end{bmatrix} \equiv \begin{bmatrix} p_{r1}, \ldots, p_{rk_r} \\ q_{r1}, \ldots, q_{rk_r} \end{bmatrix}. \tag{9}$$

The master problem E^k is then obtained from (1) and (2) by substituting x_r, $r=1, \ldots, R$ with convex combinations of the proposals, which by definition are feasible in their respective block constraints (3) and (4). Hence:

$$E^k \quad \begin{cases} \text{maximize } z^k \equiv \sum_{r=1}^{R} p_r^{k_r} \lambda_r^{k_r}, & (10) \\ \text{subject to } \sum_{r=1}^{R} Q_r^{k_r} \lambda_r^{k_r} = d_0, & (11) \\ \delta_r^{k_r} \lambda_r^{k_r} = 1; \quad r=1, \ldots, R, & (12) \\ \lambda_r^{k_r} \geqq 0; \quad r=1, \ldots, R, & (13) \end{cases}$$

where $\lambda_r^{k_r} \equiv (\lambda_{r1}, \ldots, \lambda_{rk_r})$ with λ_{rl} being the weight on the l-th proposal from block r; and $\delta_r^{k_r} \equiv (\delta_{r1}, \ldots, \delta_{rk_r})$ with δ_{rl} being one if the l-th proposal corresponds to an extreme solution, and zero otherwise. Finally, denote the vector of dual variables in E_k by $(\pi_0^k, \sigma_1^k, \ldots, \sigma_R^k)$ where π_0^k corresponds to the coupling constraints (11) and σ_r^k to the r-th convexity constraint in (12).

The algorithm then consists of three phases. Phase 1 seeks proposals from the subproblems to make the master problem feasible. Phase 2 generates proposals to optimize the master problem. Phase 3 reconstructs an optimal solution to the original problem.

Computational strategies are refinements of the basic algorithm aimed at accelerating convergence, maintaining numerical stability and reducing memory as well as data handling requirements. Our experience indicates that the following are crucial to an efficient implementation.

(i) Partial cycles: in which only certain active subproblems are selected for solution;

(ii) Multi-proposal generation: to produce, if possible, more than one proposal from a subproblem;

(iii) Proposal purging: to keep memory requirement manageable, the inactive proposals in the master problem must be purged periodically to make room for new proposals.

3.3. The Codes

DECOMP is a Fortran implementation of the Dantzig-Wolfe Decomposition algorithm originally coded by C. Winkler at the Systems Optimization Laboratory, Stanford University. It was based on J. Tomlin's LPM1 linear

programming code [9]. It has since been modified and extended by the authors at CORE [8]. When dimensioned for a maximum of 10 subproblems, each with up to 400 rows, 1,000 columns and 6,000 nonzero coefficients, and a maximum of 99 coupling constraints, and when compiled with IBM FORTRAN IV Extended with optimization level 2, DECOMP requires about 230 K bytes of core storage and 106 records of 13,028 bytes on a direct access device (e. g. disk). Since it is coded in Fortran, DECOMP is, on the whole, portable (for example it has been adapted for a Data General MV8000 computer).

DECOMPSX is a decomposition code developed by the authors at CORE and based on IBM's LP software package MPSX/370. Its design, which exploits the modularity feature of MPSX/370 to a great extent, is described in Ho and Loute [6]. Unfortunately, some minor modifications of MPSX/370 were necessary so that DECOMPSX is not directly portable even given a MPSX/370 environment. When run under VM/CMS on an IBM 370/158 it uses a 1,000 K virtual machine and allows for up to 99 subproblems with a maximum of 1,000 coupling constraints. Each subproblem could theoretically have up to 16,000 rows, but a practical limit is in the range of 2,000-4,000 rows. At this point we observe that both DECOMP and DECOMPSX can easily accommodate the yearly DYNAMICO model described in the previous section.

4. COMPUTATIONAL RESULTS

In this section, the experience of solving a typical case of DYNAMICO using DECOMP and DECOMPSX is reported. In both experiments, each of the ten submodels is optimized and the proposal corresponding to maximal regional GNP is used to initiate the master problem which obviously would not yet be feasible. The programs then automatically went through what is equivalent to a Phase 1 of the simplex method to achieve feasibility for the master problem. Recall that Costa and Jennergren [3] has to resort to certain "tricks" to generate the first 15 prelink proposals from each submodel that would make the master problem feasible. Here, such logical intervention is not necessary although it might help to improve convergence. After feasibility of the master problem is attained, the programs entered a Phase 2 to maximize the gross world product and terminates when the value of the maximand is within a given relative tolerance of its upper bound. Finally, the programs proceeded to a Phase 3 procedure to reconstruct a primal optimal solution to each submodel.

DECOMP was run on a Data General MV8000 under AOS/VS. The convergence tolerance used was 0.1%. The actual tolerance achieved was 0.04%. The average time for the first optimization of a submodel was 2.8 seconds. The average time per iteration was 13.6 seconds. The statistics for this run are given in table 1. We also note here that if the convergence tolerance was set to 1%, DECOMP required a total of 6 iterations and 165.48 seconds achieving an actual tolerance of 0.5%.

TABLE 1

Solution statistics of DECOMP

DECOMP	Cycles	No. of Proposals	CPU Time (sec.)
Input + opt. of submodels	–	10	62.36
Phase 1	4	145	53.08
Phase 2	5	144	69.28
Phase 3 + output	–	–	19.33
TOTAL	9	299	204.05

DECOMPSX was run on an IBM 370/158 under VM/CMS using a 1,000 K virtual machine. The tolerance used was 0.1%. The statistics are in table 2, where for comparison, we have also listed those for applying MPSX/370 directly to the entire problem.

TABLE 2

Solution statistics of DECOMPSX

	MPSX/370		DECOMPSX	
	Iterations	CPU (sec.)	Cycles	CPU (sec.)
Phase 1	753	155.4	7	144.8
Phase 2	217	76.6	7	99.4
Phase 3	–	–	–	43.8
TOTAL	970	232.0	14	288.0

The relatively low number of cycles (9 for DECOMP and 14 for DECOMPSX) required indicates that the model is a very well behaving one for decomposition. Although we have only shown the performance statistics for the base case, similar results hold for other scenarios.

There is another experience we gained from this experiment that brought forth the advantage of the decomposition approach. It turned out that certain errors (beyond our control) had been introduced in the process of transforming the original data into the format for DECOMP and DECOMPSX. As a consequence, various subproblems became infeasible. Using decomposition, we were able to analyze these one at a time and detect the errors much more expeditiously than having to deal with the entire problem all at once.

5. CONCLUDING REMARKS AND FUTURE DIRECTIONS

Our results, though very limited, do show that DYNAMICO can be much improved as a software package for policy modeling if advanced computational techniques in decomposition are incorporated. Note that both the computing times reported for the Data General MV8000 and IBM 370/158 translate to under 1 minute for an IBM 370/168.

Among the possibilities of future development are:

(1) routine operation of the actual DYNAMICO model at significantly reduced computer and human resources by incorporating an *interactive* version of e. g. DECOMP;

(2) expansion of the model to include more details of regional economies;

(3) reformulate the model as a truly dynamic optimization model to allow the study of optimal long range development policies.

The last two points are based on the observation that using software like DECOMP and DECOMPSX, one can solve much larger models than the actual yearly DYNAMICO within the same computational environment.

ACKNOWLEDGMENTS

The computations reported were performed at the Center for Operations Research and Econometrics. Partial support for the first author was provided by the College of Business Administration, The University of Tennessee; the Faculté des Sciences Appliquées, Université Catholique de Louvain; and the College Interuniversitaire d'Études Doctorales dans les Sciences du Management. He is also grateful to Antonio Costa formerly of the United Nations Secretariat for providing the model and many stimulating discussions. The presentation has benefitted greatly from the very helpful comments and suggestions of an anonymous referee.

REFERENCES

1. M. Benichou, J. M. Gauthier, G. Hentges and G. Ribiere, *The Efficient Solution of Large-Scale Linear Programming Problems—Some Algorithmic Techniques and Computational Results*, Mathematical Programming, Vol. 13, 1977, pp. 280-332.

2. V. CHVÁTAL, *Linear Programming*, W. H. Freeman, New York, 1983.
3. A. M. COSTA and L. P. JENNERGREN, *Trade and Development in the World Economy: Methodological Features of Project DYNAMICO*, Journal of Policy Modeling, Vol. 4, 1982, pp. 3-22.
4. G. B. DANTZIG and P. WOLFE, *The Decomposition Algorithm for Linear Programs*, Econometrica, Vol. 19, 1961, pp. 767-778.
5. Y. M. I. DIRICKX and L. P. JENNERGREN, *Systems Analysis by Multilevel Methods: With Applications to Economics and Management*, Wiley, Chichester, England, 1979.
6. J. K. Ho and É. LOUTE, *An Advanced Implementation of the Dantzig-Wolfe Decomposition Algorithm for Linear Programming*, Mathematical Programming, Vol. 20, 1981, pp. 303-326.
7. J. K. Ho and É. LOUTE, *Computational Experience with Advanced Implementation of Decomposition Algorithms for Linear Programming*, Mathematical Programming, Vol. 27, 1983, pp. 283-290.
8. J. K. Ho and É. LOUTE, *DECOMP User's Guide*, unpublished manuscript.
9. J. A. TOMLIN, *LPM1 User's Guide*, unpublished manuscript.