Efficient Management of Multiple Sets to Extract Complex Structures from Mathematical Programs *

Emmanuel Fragnière [a] Jacek Gondzio [b] Robert Sarkissian [c]

[a] *HEC, Department of Management, University of Lausanne, BFSH1, 1015 Dorigny-Lausanne, Switzerland, e-mail: Emmanuel.Fragniere@hec.unil.ch*

[b] *Logilab, HEC, Department of Management, University of Geneva, CH-1205, Switzerland, e-mail: Jacek.Gondzio@hec.unige.ch*

[c] *Judge Institute of Management Studies, Cambridge University, Trumpington Street, Cambridge, England, CB2 1AG*

Most of the applied models written with an algebraic modeling language involve simultaneously several dimensions such as materials, location, time or uncertainty. The information about dimensions available in the algebraic formulation is usually sufficient to retrieve different block structures from mathematical programs. These structured problems can then be solved by adequate solution techniques. To illustrate this idea we focus on stochastic programming problems with recourse. Taking into account both time and uncertainty dimensions of these problems, we are able to retrieve different customized structures in their constraint matrices. We applied the Structure Exploiting Tool to retrieve the structure from models built with the GAMS modeling language. The underlying mathematical programs are solved with the decomposition algorithm that applies interior point methods. The optimization algorithm is run in a sequential and in a parallel computing environment.

Keywords: Algebraic modeling language, large-scale optimization, structure exploiting solver, stochastic programming with recourse.

* The research of the first author was supported by the Fonds National de la Recherche Scientifique Suisse, grant #1214-049696.96/1. The research of the second author was supported by the Fonds National de la Recherche Scientifique Suisse, grant #12-42503.94.

1. Introduction

Nowadays economic models involve relations defined over several dimensions such as location, materials, time or uncertainty. They are often built and translated into a Mathematical Program (MP) with an Algebraic Modeling Language (AML) [4,6,15,24]. As a matter of fact it is quite natural for the economist to write models with equivalent algebraic notations. Dimensions are in this context represented by the use of indexed sets in the algebraic formulation. Unfortunately, when generating the complete MP, algebraic modeling languages lose structures that were easily identifiable in the algebraic formulation of a given model. We show in this paper that by an efficient management of multiple indexed sets, within an algebraic modeling language, it is possible to extract easily customized block structures.

Extracting complex block structures from an unstructured mathematical program is a difficult task. It is however a mandatory step to exploit them with adequate algorithmic techniques. Several heuristics exist that can detect automatically certain special forms. Although interesting work has been undertaken (see for instance [13]), detecting a block structure from an unstructured MP remains an NP complete problem. Moreover, in case of very complex forms such as nested block structures, some doubt can be raised about their potential efficiency. If the cost incurred by the extraction of special forms exceeds the gains obtained to exploit them, then this approach is not attractive anymore. An alternative is to refer directly to the "semantic" of the original model.

We focus here on multistage stochastic programming problems with recourse since they combine two main dimensions: time and randomness. We should insist that we could have chosen other problems as an illustration. The work on both dimensions allows us to customize precisely the shape of the desired structure (e.g., staircase structure, simple or nested dual block-angular structures) and the sizes of blocks involved). It is indeed difficult to determine in advance which shape benefits at best from a given structure exploiting solver. In order to extract and exploit different block structures within the algebraic modeling language, we employ the concept SET [16] that stands for Structure Exploiting Tool. SET is made up of two distinctive parts. SPI (Structure Passing Interface) extracts the

structure desired by the modeler. Then SES (Structure Exploiting Solver) is an adequate solver that takes advantage of the extracted form.

The SES that we hooked to GAMS and used to solve stochastic programming problems is an interior-point based decomposition method [23]. The master problem is solved by the Analytic Center Cutting Plane Method (ACCPM) [22]. The subproblems are solved with HOPDM LP code [21]. Experiments are both realized in sequential and parallel computing environments.

Finally, let us mention that there exist optimization tools such as as DEC IS [26], MsLip [20] and SP/OSL [27] dedicated to stochastic programming that take advantage of the slight differences among scenarios to avoid data redundancy. We have thus used SET to connect a specialized stochastic optimization solver to an algebraic modeling language. We think that this approach could complement recent attemps to write multistage stochastic programs directly in AMLs.

This paper is organized as follows. In Section 2, we explain the links that exist between the multistage stochastic programming problem and nested dual block-angular structures. In Section 3, we recall the concept SET that is used for the first time to extract nested block structures. We also show that different structures can be extracted from the same model by manipulating multiple sets. In Section 4, we examine several test problems generated by GAMS and solved by an IPM-based decomposition method both on an IBM RS/6000 machine and on a cluster of PC's. In Section 5, we show a way of linking GAMS and SP/OSL through the use of the concept SET. In Section 6, we conclude this paper by indicating several research directions that could be initiated from this work.

2. Nested structures in stochastic programs

Modeling languages provide very expressive environment because they are close to the mathematical model and offer connection to advanced optimization techniques granted by some commercial solvers. Complex models can be formulated with generic constraints and variables depending on several dimensions (e.g. location, time, product line, region). Stochastic programming models involve, for instance, time and uncertainty. Initiated in the late fifties by Dantzig and Madan-

sky [8], stochastic programming is a growing field of mathematical programming. The following books provide a good overview of the field: [3,9,12,26,32].

2.1. The two-stage problem

We first consider the following two-stage stochastic linear program

$$\begin{aligned} \min \quad & E_{\tilde{\xi}}\{c'x + Q(x,\tilde{\xi})\} \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0, \end{aligned} \tag{1}$$

where $\tilde{\xi}$ is a random vector and Ξ is the set of all possible ξ, $x \in R_+^{n_1}$ is the vector of first stage variables, $A \in R^{m_1 \times n_1}$ is the first stage constraint matrix, $b \in R^{m_1}$ is the first stage right hand side vector, $c \in R^{n_1}$ is the objective function vector of the first stage. $Q(x,\xi)$ is the optimal value of another linear program; it is a function of the first stage decisions x and of the realization $\xi \sim (q(\xi), h(\xi), T(\xi))$

$$Q(x,\xi) = \min\{\ (q(\xi))'y \mid Wy = h(\xi) - T(\xi)x, \quad y \geq 0\ \}, \tag{2}$$

where $y \in R_+^{n_2}$ is the vector of second stage variables, $T(\xi) \in R^{m_2 \times n_1}$ is a realization of the random matrix which links first and second stages, $W \in R^{m_2 \times n_2}$ is the technology matrix of the second stage (we assume that this matrix is deterministic). W could also be a random matrix in a more general formulation of the problem (1-2). By assuming that W is a deterministic matrix the problem belongs to the class of stochastic problems *with fixed recourse*. Moreover if the rank of matrix W is equal to m_2 the problem has *complete fixed recourse*. It implies that whatever decision x and realization ξ the second stage problem $Q(x,\xi) = \min\{(q(\xi))'y | Wy = h(\xi) - T(\xi)x, \quad y \geq 0\}$ is feasible.

The problem (1) is called the *first stage problem*. Decision x must be taken before the realization ξ can be observed. The problem (2) is called the *second stage problem* or the *recourse problem*. After the true environment is observed, differences that can appear between $h(\xi)$ and $T(\xi)x$ (for x given and for $h(\xi)$ and $T(\xi)$ observed) are corrected by determining a recourse action $y \geq 0$ that satisfies the following relation $Wy = h(\xi) - T(\xi)x$, and that minimizes the cost $(q(\xi))'y$. To sum up, the aim of the two-stage problem is to find for the problem

(1-2) a solution x which is feasible for all realizations ξ and which minimizes the expected cost. We note the adaptative nature of this model.

Let us consider the case where the distribution of the random vector $\tilde{\xi}$ is discrete $\xi^l = (q^l, h^l, T^l)$ with probability $p^l \geq 0$, $l = 1, 2, \ldots, L$, where $\sum_{l=1}^{L} p^l = 1$. The two-stage problem (1-2) becomes

$$\begin{aligned} \min\ & c'x + \sum_{l=1}^{L} p^l Q(x, \xi^l) \\ \text{s.t. } & Ax = b \\ & x \geq 0, \end{aligned} \tag{3}$$

where $Q(x, \xi^l)$, $l = 1, 2, \ldots, L$, is the optimal objective function of the recourse problem

$$\begin{aligned} Q(x, \xi^l) = \min \quad & (q^l)'y \\ \text{s.t.} \quad & Wy = h^l - T^l x, \\ & y \geq 0. \end{aligned} \tag{4}$$

For a given x, let $\hat{y}^l(x)$ be the optimal solution of (4) for $l = 1, 2, \ldots, L$. The objective function of (3) becomes $c'x + \sum_{l=1}^{L} p^l (q^l)'\hat{y}^l(x)$. We are now left with a deterministic equivalent

$$\begin{array}{lllllll} \min & c'x & +p^1(q^1)'y^1 + p_2(q^2)'y^2 + \ldots + p_L(q^L)'y^L & & & & \\ \text{s.t.} & & & & & & \\ & Ax & & & & & = b \\ & T^1x & +Wy^1 & & & & = h^1 \\ & T^2x & & +Wy^2 & & & = h^2 \\ & \vdots & & & \ddots & & \vdots \\ & T^Lx & & & & +Wy^L & = h^L \\ & x \geq 0, & y^1 \geq 0, & y^2 \geq 0, & \ldots & y^L \geq 0. & \end{array} \tag{5}$$

We note that the constraint matrix of this problem displays a dual block-angular structure.

2.2. The multi-stage problem

In a planning approach the evolution of uncertainties can be described as an alternation of decisions and random realizations. The generalization of the two-stage problem when we have to deal with several periods, leads to the multi-stage problem. By extending the two-stage formulation, we obtain the following multi-stage formulation

$$
\begin{array}{lllll}
\min & c'x + \displaystyle\sum_{l_2=2}^{L_2} p^{l_2}(q^{l_2})'y^{l_2} + \sum_{l_3=L_2+1}^{L_3} p^{l_3}(q^{l_3})'y^{l_3} + \ldots + \sum_{l_T=L_{T-1}+1}^{L_T} p^{l_T}(q^{l_T})'y^{l_T} & & & \\
\text{s.t.} & & & & \\
 & Ax & & = b & \\
 & T^{l_2}x + W^2 y^{l_2} & & = h^{l_2}, & l_2 = 2,\ldots,L_2, \\
 & T^{l_3}y^{a(l_3)} + W^3 y^{l_3} & & = h^{l_3}, & l_3 = L_2+1,\ldots,L_3, \\
 & \ddots & & \vdots & \\
 & T^{l_T}y^{a(l_T)} + W^T y^{l_T} & & = h^{l_T}, & l_T = L_{T-1}+1,\ldots,L_T, \\
 & x \geq 0, \quad y^{l_t} \geq 0, & l_t = 2,3,\ldots,L_T, & &
\end{array}
$$

where t denotes the stage, l_t is the index of the realization of stage t, L_t is the last realization belonging to stage t, $a(l_t)$ corresponds to the direct ancestor of realization l_t.

Two sets describe the main dimensions in the multistage stochastic programming (the time and the randomness). The different subsets at each stage differs in size in respect to probability distributions of the appropriate stochastic process.

2.3. Nested dual block-angular structure

In its simplest form the discrete stochastic process can be represented as an event tree describing the unfolding of the uncertainty over the period of planning (see Figure 1).

A path, from the root to a leaf of the event tree, represents a scenario. Each scenario has a given probability. At each node of the event tree, a set of constraints and an economic function, which involves variables specific to that node and its predecessor nodes, are defined. For instance, node 1 of the tree presented in Figure 1 corresponds to the first stage and associated decisions are identical for scenarios 1, 2, 3 and 4. At stage 2, decisions of scenarios 1 and 2

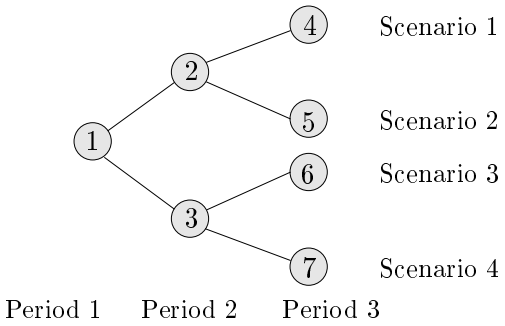

Figure 1. A simple event tree

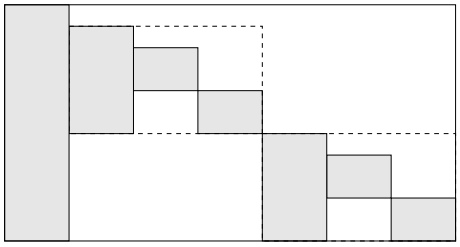

Figure 2. The constraint matrix associated with the event tree 1

are identical. In the same way decisions of scenarios 3 and 4 are identical. If the ordering of the time and randomness dimensions is maintained during the generation of the mathematical program, the corresponding constraint matrix (shown in Figure 2) displays a nested dual block-angular structure. Links between the nested dual block-angular structure and the algebraic formulation of the original model can be easily established.

In our example (see Figure 1), node 1 is the root node, nodes 2 and 3 belong to stage 2 ($l_2 = 2, 3$), nodes 4 to 7 belong to stage 3 ($l_3 = 4, \ldots, 7$). Its equivalent algebraic formulation is presented in (6). Let us note that by shifting y_3 just after y_5, we immediately identify the shape that was presented in Figure 2.

$$
\begin{array}{lllllllll}
\min & c'x & +p^2(q^2)'y^2 + p^3(q^3)'y^3 + p^4(q^4)'y^4 + p^5(q^5)'y^5 + p^6(q^6)'y^6 + p^7(q^7)'y^7 & & & & & & \\
\text{s.t.} & & & & & & & & \\
 & Ax & & & & & & & = b \\
 & T^2x & + W^2y^2 & & & & & & = h^2 \\
 & T^3x & & + W^2y^3 & & & & & = h^3 \\
 & & T^4y^2 & & + W^3y^4 & & & & = h^4 \\
 & & T^5y^2 & & & + W^3y^5 & & & = h^5 \\
 & & & T^6y^3 & & & + W^3y^6 & & = h^6 \\
 & & & T^7y^3 & & & & + W^3y^7 & = h^7 \\
 & x \geq 0, & y^2 \geq 0, & y^3 \geq 0, & y^4 \geq 0, & y^5 \geq 0, & y^6 \geq 0, & y^7 \geq 0. &
\end{array}
\tag{6}
$$

Apparently, (6) represents a structured linear program. Its structure should be explored in the solution algorithm. Unfortunately, when the model is written with an algebraic modeling language the structure, easily identifiable in the algebraic formulation, is usually lost when the corresponding mathematical program is sent to the solver. Each algebraic modeling language uses its own algorithm to generate an equivalent mathematical program, which scrambles the structure. Consequently, the structure of stochastic program, though existing in the model, is unrecognizable for the solver. In the following section we briefly recall SET, a tool that enables to extract structures from algebraic modeling languages: we use it to extract dual block-angular structures from multistage stochastic programs.

3. Block structures of stochastic programs

The model considered in this section is a simple financial planning problem suggested by Birge. At each period, one can invest in 4 securities and cash. There are no transaction costs. The initial capital and the desired wealth at the end of the planning period are defined exogenously. Prices of securities have been computed by a multivariate log-normal random generator implemented in MATLAB. Prices are assumed independent between periods. The time-scale dimension is defined over T periods. The uncertainty dimension is defined at each node for N realizations. The corresponding event tree is symmetric and involves N different branches at each node except for the leaves. Hence the total number of scenarios is N^T. The objective is a piecewise linear function (a

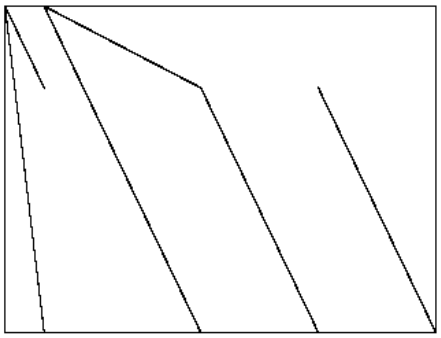

Figure 3. Jacobian matrix generated by GAMS

scenario is strongly penalized when the goal is not reached), which corresponds to the expected utility of all scenarios.

The model is written in the GAMS modeling language. We use the structure exploiting tool, SET to retrieve the structure from the algebraic modeling language. This is done in two steps. First, structure passing interface, SPI, retrieves the desired structure from the unstructured mathematical program built by the algebraic modeling language. Second, SPI passes the special structure of the problem to a SES (structure exploiting solver).

Figure 3 shows an example of the constraint matrix generated by GAMS. No apparent block structure can be identified from this matrix since GAMS uses its own ordering scheme. Then passing a few information about time and uncertainty dimensions SPI produces directly from GAMS a nested dual-block angular structure as shown in Figure 4.

The presence of more than one dimension opens the possibility of retrieving different structures from the same model. This is in particular the case in stochastic programming problems. Ideally, one should be able to customize the structure for a specific solver. Decomposition approaches, for instance, can take advantage of the presence of large number of comparably difficult subproblems and a small dimension of the master problem. Other structure exploiting solvers can benefit from a particular sparsity structure of sub-blocks involved. We shall address these issues in more detail in the next section.

Let us return to the financial planning model. We have already presented the most obvious reordering of this stochastic program based on both time and

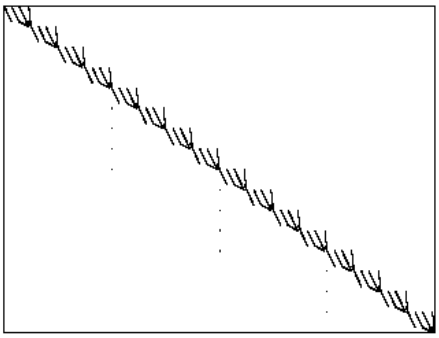

Figure 4. Reordered Jacobian matrix displaying a nested dual block angular structure

uncertainty dimensions. Let us explore different choices.

By playing solely with the randomness dimension and restricting to the first-stage decisions, it is possible to extract a dual block-angular structure as shown in Figure 5. First stage coefficients belong to the linking block (due to the small number of first stage variables they are invisible in Figure 5). Each scenario implied by the realization of the first stage variable defines a separate aggregated second stage block.

Starting from the structure displayed in Figure 4, by shifting the columns of the second stage next to those of the first stage (as shown in Figure 6), we obtain a dual block-angular structure with a higher granularity (as shown in Figure 7).

Alternatively, by restricting our analysis solely to time dimension, we can produce a staircase structure exhibited in Figure 8. We limited the number of events to make the structure more apparent.

Producing desired block structures from an algebraic modeling language enables us to access particular structure exploiting solvers; in the next section we show its relevance to the efficiency of the solver.

4. From model to solution

We apply SET to produce dual block-angular structures from stochastic linear programs formulated with GAMS modeling language. Dual block-angular structure is well suited to the Benders decomposition [2,20,27]. The decomposition method we use belongs to this class. The Analytic Center Cutting Plane

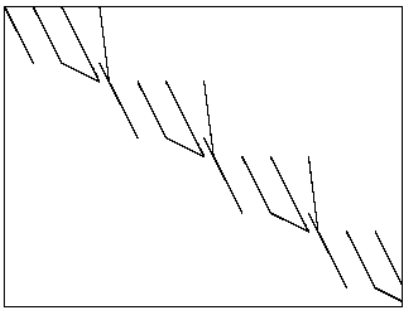

Figure 5. Dual block angular structure

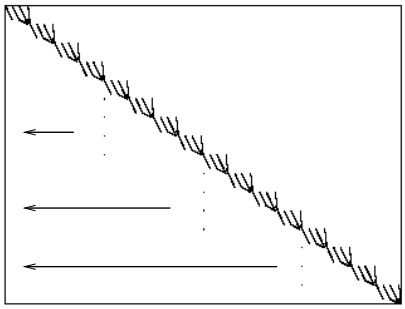

Figure 6. Moving columns from the second stage to the first stage

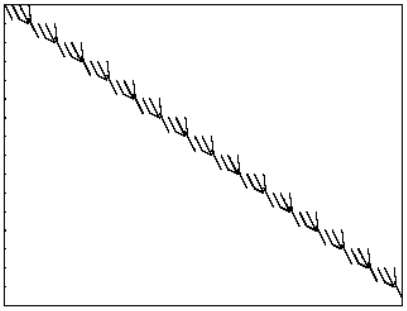

Figure 7. Resulting dual block angular structure

Method solves the restricted master problem and HOPDM solves the subproblems. Consequently, we apply interior point methods at both levels of decomposition.

Our choice of the solution technique was essentially motivated by the pres-

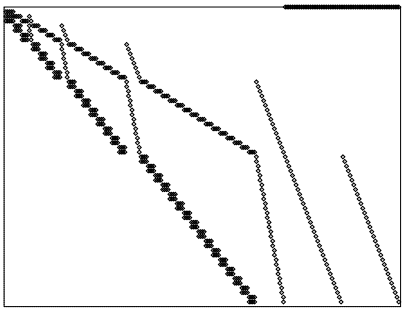

Figure 8. Staircase structure

ence of the appropriate tools in our laboratory. We aimed to show that SET is a reliable tool to exploit structures of large scale mathematical programs generated directly from the algebraic modeling language rather than looking for the most efficient solver adjusted to a given class of problems. For instance, we could have hooked any algorithm that exploits staircase structure or any decomposition method dedicated to stochastic optimization problems.

We report numerical experiments for the financial planing model presented in Section 3. The sequential decomposition code was run on an IBM RS/6000, whereas its parallelized version was run on a cluster of 10 Pentium Pro PC's.

4.1. Sequential decomposition

In Table 1, we report the characteristics of the problems as well as the size of the corresponding linear programs. For instance, problem P6R7 has 6 periods (i.e. 7 stages in stochastic programming terminology) and 7 realizations at each node of the event tree. Consequently, the event tree has $7^6 = 117649$ scenarios.

In Table 2, we report solution statistics. Our experiments were conducted on an IBM RS6000 (58MHZ Power2 processor with 128MB of memory). Each problem was decomposed in two different ways. To begin with, the event tree was cut between the first and the second stage (see Figure 5). As the event tree is symmetrical, such partitioning results in subproblems of identical sizes. The number of subproblems is equal to the number of realizations in the first stage. Next, the event tree is cut between the second stage and the third stage (see Figure 7) leading thus to p^2 subproblems where p is the number of realizations.

The overall solution time includes

- the time spent in GAMS for model generation,
- the time of reading GAMS dictionary and generating the desired partitioning with SPI,
- the time spent in the decomposition algorithm,
- the time to send the solution in the original order back to the AML.

In Table 2, we report only the second and the third one and their sum. We also report the solution time of these problems with the general-purpose linear solver OSL from IBM [25].

Table 1
Description of problems generated with GAMS.

Problem	Periods	Events	Scenarios	Rows	Columns
P6R3	6	3	729	1094	3279
P6R4	6	4	4096	5462	15018
P6R5	6	5	15625	19532	50781
P6R6	6	6	46656	55988	139968
P6R7	6	7	117649	137258	338339

The overall execution time of problem P6R7 (with 7 subproblems) is less than one hour. This shows that large models can be generated with a modeling language and solved efficiently on a single machine when one uses the structure exploiting solver. We observe that compared with the direct approach, the decomposition is more attractive for larger problems. Direct solution of the largest problem P6R7 could not complete in a reasonable CPU time. Moreover it appears that cutting the event tree between stages 1 and 2 rather than between stages 2 and 3 is more interesting for this particular problem. The problem P6R7, decomposed into 49 supbroblems, could not be solved due to excessive storage requirements.

Table 2
CPU times in seconds, IBM RS6000

Problem	Subproblems	Partitioning	Decomposition	Total	OSL
P6R3	3	5	14	19	6
P6R4	4	31	56	87	155
P6R5	5	126	212	338	1803
P6R6	6	403	535	938	16018
P6R7	7	1099	2312	3411	-
P6R3	9	14	18	32	6
P6R4	16	104	69	173	155
P6R5	25	533	256	789	1803
P6R6	36	2886	664	3550	16018
P6R7	49	-	-	-	-

4.2. Parallel decomposition

Decomposition is an algorithmic device that breaks down computations into several independent subproblems. It is thus ideally suited to parallel implementation. We have repeated the computations on a parallel machine. We used a cluster of 10 Pentium Pro PC's [30] linked with fast Ethernet that allows a 10MB/s transfer rate. One machine handles the master problem, whereas the nine slave machines handle the subproblems. The operating system is LINUX and the parallel communication is done with MPI [5].

We illustrate the behavior of the parallel decomposition on one problem P6R4. This problem has 4 realizations at each node of the event tree. It can naturally be decomposed either into 4 or into 16 subproblems depending on where the complete model is broken. We ran these two instances on 1,2,4 and 8 processors of the parallel machine and collected the solution times in Table 3. We could notice reasonable speed-ups increasing linearly with the number of processors. The main reason is that subproblems have exactly the same size which facilitates load balancing [23]. Specific issues of the parallel interior point decomposition of huge stochastic programs will be addressed in more detail in a subsequent paper.

Table 3
CPU times in seconds, cluster of 10 PC's

Problem	1 Processor	2 Processors	4 Processors	8 Processors
P6R4-4	54.71	34.94	19.58	not run
P6R4-16	76.80	41.77	26.05	15.68

5. Another approach: S-MPS based codes

In the previous section we presented multistage programs that were generated from an algebraic modeling language. In particular, we showed that SET enables to retrieve complex block structures from models that include several dimensions. We focused on multistage stochastic programs since they contain two main dimensions: time and uncertainty. We exploited both these dimensions when looking for the most favorable structure for decomposition.

By contrast, the S-MPS format [1] separates the time and randomness dimensions. It extends the MPS file using three files: the "core" file, which closely follows the MPS format and contains the basic deterministic problem; the "time" file which explicitly defines the problem's time structure; and the "stoch" file, which defines the structure of the event tree and random variables and parameters. The main advantage of this format is to avoid data redundancy and, in consequence, useless memory allocation since usually scenarios differ very slightly.

Consequently, the AML does not have to produce the excessively large deterministic equivalent of the stochastic program if it is to communicate with an S-MPS based solver. Information describing the core model is prepared in a proper format: additionally, each coefficient of the constraint matrix is associated to a given stage. To show the reliability of this idea we have used SET to connect SP/OSL [27] stochastic optimization solver to GAMS [6] modeling language.

We illustrate this development with a small example, *Invendam* [31] which is a multiperiod inventory model. The objective function corresponds to the net profit. There are three generic variables (inventory, quantity bought, quantity sold) and one generic constraint (inventory balance), all indexed over the time. The model is generated for 5 periods. Figure 9 shows the constraint matrix of the "core" model as generated by GAMS. Then using SPI we produce a staircase

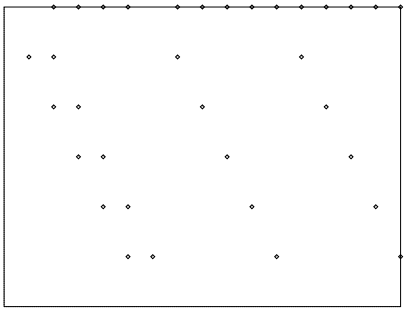

Figure 9. Jacobian matrix of the Invendam problem generated by GAMS

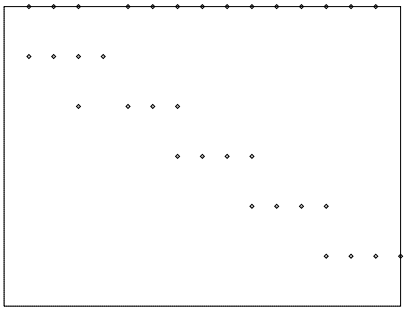

Figure 10. Staircase structure produced by SPI

structure as shown in Figure 10. Such a core model is read by SP/OSL. Stochastic parameters (e.g., event tree, probability distribution) can be defined easily by using SP/OSL features. The stochastic program produced by SP/OSL is then solved by the Benders decomposition. Although our goal was not initially to deal with the S-MPS format, we have noticed that combining the concept SET with the S-MPS format offers some advantages. The user can indeed continue to write models with an algebraic modeling language and have an easy access to codes based on the S-MPS format. This approach could perhaps complement recent developments to include in AML the possibility of writing multistage stochastic programs [10,11,14,18,19,28]

6. Conclusion

Algebraic modeling languages often loose important information the solver could have taken advantage of. SET (Structure Exploiting Tool) [16] is a new framework for retrieving this information directly from the algebraic formulation of models written in a modeling language. The main contribution of this paper is to show that by an efficient management of multiple sets it is possible to extract different kinds of block structures from the same model. In this paper we focused on multistage stochastic programming problems since they contain two main dimensions: time and uncertainty. We were thus able to produce staircase block structures retaining solely the time dimension; simple and nested dual block-angular structures, by taking into account simultaneously the time and uncertainty dimensions. The resulting structure can be exploited by an appropriate solver.

We tested an IPM-based decomposition for dual block-angularly structured linear programs. This task would have been tedious if we had not used SET to interface the AML and the solver. Our experiments confirmed the well-known fact that exploiting the structure with an adequate solver offers computational advantages.

For instance, one of the LP's has 137,258 equations and 338,339 variables, and is solved in about one hour (including the time spent in the generation of the LP by GAMS, its partitioning, and its solution by the decomposition algorithm). Moreover, general purpose solvers (hooked to the GAMS modeling language) failed to solve this problem in an acceptable time. We also demonstrated that the use of SET gives to the modeling language an access to parallel optimization techniques: we have performed our experiments both on sequential and parallel computing environments.

Due to the excessive data redundancy in stochastic optimization problems, generating the whole deterministic equivalent LP, with algebraic modeling language, seems a poor choice. The S-MPS format [1] reduces data redundancy. As long as AMLs do not include the facility to model stochastic optimization problems and simultaneously, access the structure exploiting solvers, SET seems a valuable choice. SET can be used to produce the core model (the baseline sce-

nario) dynamically ordered from the modeling language keeping uncertainty information outside AML. It thus complements recent tendencies [10,11,14,18,19,28] to integrate features into the algebraic modeling language in order to write stochastic programming problems. We showed the feasibility of this approach connecting SP/OSL to GAMS.

Finally, we should again insist that we concentrated on stochastic programming just because it is a class of problems that particularly well illustrate the idea of generating different block structures from the same model. Naturally, these developments could be useful in retrieving interesting block structures from other kinds of models emerging from fields such as telecommunications [29], energy and environment [17]. Another important point is that this technology can also be adapted to exploit structures associated with nonlinear programming problems generated by the algebraic modeling language as described in [7].

Acknowledgment

We are grateful to our colleague from Logilab, Cristian Condevaux-Lanloy for linking SP/OSL to SET.

References

[1] J. Birge, M. Dempster, H. Gassmann, E. Gunn, and S. W. A.J. King, *A standard input format for multiperiod stochastic linear programs*, Committee on Algorithms Newsletter, 17 (1988), pp. 1–19.

[2] J. R. Birge, *Decomposition and partitioning methods for multistage stochastic linear programs*, Operations Research, 33 (1985), pp. 989–1007.

[3] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming*, Springer Series in Operations Research, Berlin, 1997.

[4] J. Bisschop and R. Entriken, *AIMMS: The modeling system.* Paragon Decision Technology, 1993.

[5] P. Bridges, N. Doss, W. Gropp, E. Karrels, E. Lusk, and A. Skjellum, *User's Guide to MPICH, a Portable Implementation of MPI*, Argone National Laboratory, September 1995.

[6] A. Brooke, D. Kendrick, and A. Meeraus, *GAMS: A User's Guide*, The Scientific Press, Redwood City, California, 1992.

[7] D. Chang and E. Fragnière, *Splitdat and decomp: Two new GAMS I/O subroutines to handle mathematical programming problems with an automated decomposition procedure.* Stanford University, Department of Operations Research, manuscript, August 1996.

[8] G. Dantzig and M. Madansky, *On the solution of two-stage linear program under uncertainty*, in Proc. Fourth Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, 1961, University of California Press, pp. 165–176.

[9] M. Dempster, (ed.) *Stochastic Programming*, Academic Press, New York, 1980.

[10] S. Dirkse, *GAMS/SP/OSL.* GAMS Developer's Workshop, Stuttgart, November 3-7, 1997.

[11] R. Entriken, *Algebraic language constructs for stochastic optimization.* INFORMS Conference, Atlanta, November 3-6, 1996.

[12] Y. Ermoliev and R.-B. Wets, (eds.) *Numerical techniques for stochastic optimization*, vol. 10, Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 1988.

[13] M. Ferris and J. Horn, *Partitioning mathematical programs for parallel solution*, tech. rep., Computer Sciences Department, University of Wisconsin–Madison, May 1994.

[14] R. Fourer, *Features for formulating stochastic programs in an algebraic modeling language.* INFORMS Conference, Atlanta, Novemver 3-6 1996.

[15] R. Fourer, D. Gay, and B. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, The Scientific Press, San Francisco, California, 1993.

[16] E. Fragniere, J. Gondzio, R. Sarkissian, and J.-P. Vial, *Structure exploiting tool in algebraic modeling languages*, tech. rep., Logilab, Section of Management Studies, University of Geneva, 102 Bd Carl Vogt, CH-1211 Geneva 4, Switzerland, June 1997, revised in June 1998.

[17] E. Fragniere and A. Haurie, *A stochastic programming model for energy/environment choices under uncertainty*, Int. J. Environment and Pollution, 6 (1996), pp. 587–603.

[18] H. Gassmann and A. M. Ireland, *Scenario formulation in an algebraic modelling language*, tech. rep., School of Business Administration, Dalhousie University, 1992, revised July 1994.

[19] ——, *On the automatic formulation of stochastic linear programs*, tech. rep., School of Business Administration, Dalhousie University, 1994.

[20] H. I. Gassmann, *Mslip: A computer code for the multistage stochastic linear programming problems*, Mathematical Programming, 47 (1990), pp. 407–423.

[21] J. Gondzio, *HOPDM (version 2.12) – a fast LP solver based on a primal-dual interior point method*, European Journal of Operational Research, 85 (1995), pp. 221–225.

[22] J. Gondzio, O. du Merle, R. Sarkissian, and J.-P. Vial, *ACCPM - a library for convex optimization based on an analytic center cutting plane method*, European Journal of Operational Research, 94 (1996), pp. 206–211.

[23] J. Gondzio, R. Sarkissian, and J.-P. Vial, *Parallel implementation of a central decomposition method for solving large-scale planning problems*, Tech. Rep. Logilab Technical Report 1998.1, Logilab, University of Geneva, 102 Bd Carl-Vogt, CH-1211, January 1998.

[24] T. Huerlimann, *Reference manual for the LPL modeling language (version 3.1)*. Institute for Automation and Operations Research, University of Fribourg, CH-1700, Switzerland, manuscript, 1989.

[25] IBM Corporation, *Optimization Subroutine Library Guide and References*, fourth ed., 1992.

[26] G. Infanger, *Planning under Uncertainty*, Boyd and Fraser, Massachusetts, 1994.

[27] A. J. King, S. E. Wright, and R. Entriken, *SP/OSL Version 1.0: Stochastic Programming Interface Library User's Guide*, IBM Research Division,York Heights, 1994.

[28] A. Leuba and D. Morton, *Generating stochastic linear programs in S-MPS format with GAMS*. INFORMS Conference, Atlanta, November 3-6, 1996.

[29] R. Sarkissian, *Telecommunications Networks: Routing and Survivability Optimization Using a Central Cutting Plane Method*, PhD thesis, École Polytechnique Fédérale de Lausanne, CH-1205 Ecublens, November 1997.

[30] T. Sterling, D. J. Becker, D. Savarese, J. E. Dorband, U. A. Ranawake, and C. V. Packer, *Beowulf: A parallel workstation for scientific computation*. Proceedings, International Parallel Processing Symposium, 1995.

[31] G. Thompson, *Computational Economics*, Scientific Press, 1992.

[32] S. W. Wallace and P. Kall, *Stochastic Programming*, John Wiley & Sons, Chichester, 1995.