A Planning Model with one Million Scenarios Solved on an Affordable Parallel Machine[*]

Emmanuel Fragnière[†] Jacek Gondzio[‡] Jean-Philippe Vial

Logilab, HEC, Section of Management Studies, University of Geneva

102 Bd Carl-Vogt, 1211 Geneva, Switzerland

http://ecolu-info.unige.ch/~logilab/

Logilab Technical Report 1998.11

June 29, 1998

Abstract

Stochastic programs inevitably get huge if they are to model real life problems accurately. Nowadays only massive parallel machines can solve them but at a cost few decision makers can afford. We report here on a deterministic equivalent linear programming model of 1,111,112 constraints and 2,555,556 variables generated by GAMS. It is solved by an interior point based decomposition method in less than 3 hours on a cluster of 10 Linux PC's.

Key words. Algebraic modeling language, distributed systems, financial planning, large scale optimization, structure exploiting solver.

1 Introduction

The curse of dimensionality is a major problem in optimization. To depict real life situations with greater accuracy, optimization models tend to be larger and larger, a trend that is probably encouraged by the rapid development of cheap and powerful computers. Unfortunately, hardware improvements

[*]This research was supported by the Fonds National de la Recherche Scientifique Suisse, grants #12-42503.94 and #1214-049696.96/1.

[†]HEC, Department of Management, University of Lausanne, BFSH1, 1015 Dorigny-Lausanne, Switzerland.

[‡]On leave from the Systems Research Institute, Polish Academy of Sciences, Newelska 6, 01-447 Warsaw, Poland.

are not sufficient to sastisfy the growing appetite of modelers. Each time the current size limits are pushed forward, the corresponding models, though quite detailed, are not deemed accurate enough anymore. In this paper we propose a way to handle larger stochastic programming models. The scheme uses existing software—the GAMS [7] modeling language with the SET extension [17], and the decomposition algorithm for block angular linear programs [23]—and a cheap parallel machine made of a cluster of PC's. Our implementation of decomposition algorithm uses two in-house codes ACCPM [22] and HOPDM [21]. The use of ACCPM for stochastic optimization was discussed in [1].

The literature in applied optimization reports some case studies where huge models were successfully applied in business. For instance in [27], a semi-conductor company working in close cooperation with the University of California Berkeley developed a linear programming problem to model its production system. Several models, which had had up to 160,000 constraints, were generated with a customized matrix generator and solved using IBM's Optimization Subroutine Library (OSL) software. As reported in the paper, the considerable investment in this modeling effort was worthwile since the company turned out to be profitable. However, software and hardware capabilities still seemed to have limited the model dimensions.

The inclusion of risk management techniques in models is an important motivation for creating huge optimization problems. Indeed, if exogenous parameters are to be replaced by probability distributions, the size of models explodes. Applications that involve both risk management and optimization (also known as stochastic programming, see [4, 13, 14, 24, 32] for more detail) are quite appropriate in finance. A particularly relevant example is the asset liability model for Yasuda, a Japanese insurance company, developed by the Russel Company [8]. The model contained up to 256 scenarios with an overall size of 31,062 constraints and 44,004 variables. The deterministic model was already a quite comprehensive description of Yasuda business which forced the modelers to define probability distribution with poor discretization, thereby keeping the model size to amenable level.

Still in finance, Zenios in [34, 25, 35] solved stochastic programming models with up to 130,172 scenarios corresponding to a model with 2,603,440 constraints and 18,224,080 variables on a parallel machine CM-5 with 64 processors. To our knowledge these are the largest stochastic programming models ever solved. Unfortunately, the hardware costs are very large, far beyond the budget of small businesses and/or research groups. Moreover, the necessary software adjustments to parallelize existing LP codes are sometimes complicated and expensive.

This is the reason why we have developed an integrated concept that is driven by the modeling language. This concept gives the modeler easy access (i.e., via algebraic modeling languages) to powerful and cheap distributed systems. It also opens the door to affordable experimentations in the field of large-scale optimization modeling. Using this approach we have generated the financial planning model suggested by Birge and formulated it with one million scenarios using the GAMS modeling language [7]. We used SET (Structure Exploiting Tool) [17] to extract the particular structure of this model which in this case was decomposable. Then using a parallelized interior point decomposition method, we were able to solve this problem on a cluster of 10 Linux PC's in less than three hours.

The paper is organized as follows. In section 2, we present the main idea of the decomposition algorithm and its adaptation to interior point methods. In section 3, we show how to transform an algebraic modeling language to have automatic access to a parallel machine. In Section 4, we present experiments with a financial planning model realized on a cluster of PC's. Finally we conclude this paper indicating possible implications of these developments.

2 An interior point decomposition for stochastic optimization

Decomposition is an algorithmic device that breaks down computations into several independent subproblems. It is thus ideally suited to distributed computations and to problems that are too large to be handled with a frontal approach. However, the necessary condition for a given model to be a candidate for the decomposition is that its constraint matrix presents some particular block structure (e.g., primal and/or dual block angular structures [17]). In this section we explain first the decomposition algorithm applied to the dual block angular structure which suits the structure of the financial planning model introduced in section 4. Second we show how the decomposition algorithm can take advantage of interior point techniques to solve the master and the subproblems.

To introduce the decomposition algorithm, we present the formulation of a two-stage stochastic linear program which displays a dual block-angular structure. Although there exist other block structures that could benefit from decomposition techniques, this structure corresponds to the one exhibited in our financial planning model.

2.1 The two-stage problem

We start this section from a brief summary on the stochastic optimization problem. The reader interested in more details on this type of optimization problems and various methods of solving them is referred to [4, 13, 14, 24, 32].

Consider the following two-stage stochastic linear program

$$\begin{aligned} \min \quad & E_{\tilde{\xi}}\{\langle c,x\rangle + Q(x,\tilde{\xi})\} \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0, \end{aligned} \tag{1}$$

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 outcomes $(q(\xi), h(\xi), T(\xi))$

$$Q(x,\xi) = \min\big\{\, \langle q(\xi), y\rangle \mid Wy = h(\xi) - T(\xi)x, \quad y \geq 0 \,\big\}. \tag{2}$$

In this expression $y \in R^{n_2}_+$ 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 and $W \in R^{m_2 \times n_2}$ is the technology matrix of the second stage. We assume that this matrix is deterministic. Under this assumption, the problem belongs to the class of stochastic programs *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 are decision x and outcome ξ, the second stage problem $Q(x,\xi) = \min\{\langle q(\xi), y\rangle | Wy = h(\xi) - T(\xi)x, \quad y \geq 0\}$ is feasible. In a more general formulation of the problem (1-2), the matrix W could also be random.

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 $\langle q(\xi), y\rangle$. 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.

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

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

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

$$\begin{aligned} Q(x,\xi^l) = \min \quad & \langle q^l, y\rangle \\ \text{s.t.} \quad & Wy = h^l - T^l x, \\ & y \geq 0. \end{aligned}$$

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 $c'x + \sum_{l=1}^{L} p^l \langle q^l, \hat{y}^l(x)\rangle$. We are now left with a deterministic equivalent

$$\begin{array}{llllllllll} \min & c'x & + & p^1\langle q^1,y^1\rangle & + & p_2\langle q^2,y^2\rangle & + & \ldots & + & p_L\langle q^L,y^L\rangle \\ \text{s.t.} & & & & & & & & & \\ & Ax & & & & & & & & = b \\ & T^1x & + & Wy^1 & & & & & & = h^1 \\ & T^2x & & & + & Wy^2 & & & & = h^2 \\ & \vdots & & & & & & \ddots & & \vdots \\ & T^Lx & & & & & & & + \; Wy^L & = h^L \\ & x \geq 0, & & y^1 \geq 0, & & y^2 \geq 0, & & \ldots & & y^L \geq 0. \end{array} \tag{4}$$

The constraint matrix of this problem displays a dual block-angular structure.

2.2 Benders decomposition

The use of the decomposition principle to solve the primal block-angular formulation was first suggested by Dantzig and Wolfe [12]. A dual concept, Benders' decomposition or partitioning, applicable to the dual block-angular structure formulation, was presented in [2]. In both of these methods proposals by subproblems are submitted to a master program which then solves the current linear programming approximation for new pricing information which is then transmitted to the subproblems; the latter reacts to these new prices with revised proposals, etc.

There exists a variety of decomposition methods applicable to stochastic optimization problems [3, 18, 28, 29]. In this paper we are concerned with the decomposition approach that uses interior point methods. We underline our interest in decomposition and not in a direct application of an interior point method that was also proved to be an interesting alternative [5, 25].

Consider the problem

$$\begin{array}{llll} \text{maximize} & \langle c_0, x_0\rangle & +\sum_{i=1}^{p}\langle c_i, x_i\rangle & \\ \text{subject to} & T_i x_0 & + W_i x_i = b_i & i = 1,2,\ldots,p, \\ & & x_i \geq 0, & i = 1,2,\ldots,p, \end{array} \tag{5}$$

where $c_i, x_i \in \mathcal{R}^{n_i}, i = 0, 1, ..., p,\ b_i \in \mathcal{R}^{m_i}, i = 1, ..., p$, and all matrices $T_i, i = 1, ..., p$, and $W_i, i = 1, ..., p$, have appropriate dimensions. The constraint matrix of this linear program displays a *dual* block-angular structure.

For a given x_0, let $L(x_0)$ be the optimal value of the subproblem

$$\begin{array}{lll} \text{maximize} & \sum_{i=1}^{p} \langle c_i, x_i \rangle & \\ \text{subject to} & W_i x_i = b_i - T_i x_0, & i = 1, 2, \dots, p, \\ & x_i \geq 0, & i = 1, 2, \dots, p. \end{array} \tag{6}$$

Then we can replace (5) with

$$\begin{array}{ll} \text{maximize} & \langle c_0, x_0 \rangle + L(x_0) \\ \text{subject to} & x_0 \geq 0. \end{array} \tag{7}$$

The function $L(x_0)$ is *additive*

$$L(x_0) = \sum_{i=1}^{p} L_i(x_0), \tag{8}$$

where $L_i(x_0)$, for $i = 1, ..., p$, is the value of

$$\begin{array}{ll} \text{maximize} & \langle c_i, x_i \rangle \\ \text{subject to} & W_i x_i = b_i - T_i x_0, \\ & x_i \geq 0. \end{array} \tag{9}$$

In other words, the subproblem (6) is *separable.*

Problem (9) may be infeasible for some values of x_0. Assume first that it is feasible. Then it has an optimal solution $x_i(x_0)$ with value $L_i(x_0)$. Let $\pi_i(x_0)$ be the dual optimal solution. The *optimality cut* writes

$$L_i(v) \leq L_i(x_0) + \langle T_i^T(\pi_i - \pi_i(x_0)), v - x_0 \rangle,\ \forall v.$$

Assume now that (9) is not feasible. If we assume that the dual is feasible[1], then its dual is unbounded. Let $d_i(x_0)$ be a ray along which the dual objective $\langle b_i - T_i x_0, \pi \rangle$ is unbounded, equivalently $\langle b_i - T_i x_0, d_i \rangle > 0$. Then we have the *feasibility cut*

$$\langle T_i^T d_i(x_0), v \rangle \leq 0,\ \forall v,$$

that excludes $v = x_0$ as not being part of the feasible domain of (5).

[1]Note that the dual is independent of x_0. Should the dual be infeasible, then the primal problem would be either unbounded or infeasible. We discard this pathological case.

Let us observe that the exact optimality requirement may be replaced with the ϵ-optimality; we then obtain the weaker inequality

$$L_i(v) \leq \langle T_i^T \bar{\pi}_i, v - x_0 \rangle + L_i(x_0) + \epsilon,$$

that defines a weaker LP relaxation [23].

2.3 Cutting plane methods to solve decomposed problems

In the formulation of the transformed problem, the Benders decomposition generates problems that are usually difficult despite the relatively small dimension of the variable space. The difficulty stems from the fact that the problems are essentially nondifferentiable. Their solving requires advanced algorithmic tools. In this section we concentrate on the solution method. More specifically, we present the generic cutting plane method, and its analytic center variant, to solve the nondifferentiable problem of interest.

Cutting plane methods apply to the canonical convex problem

$$\min\{f(x) \mid x \in Q \cap Q_0\}, \tag{10}$$

where $Q \subset R^n$ is a closed convex set, $Q_0 \subset R^n$ is a compact convex set, $f : R^n \mapsto R$ is a convex function.

We assume that the set Q_0 is defined explicitly, while the function f and the set Q are defined by the following *oracle*. Given $\bar{x} \in \text{int}Q_0$, the oracle answers either one statement:

1. $\bar{x}$ is in Q and there is a support vector
   $\xi \in \partial f(\bar{x})$, i.e., $f(x) \geq f(\bar{x}) + \langle \xi, x - \bar{x} \rangle$, $\forall x \in R^n$.

2. $\bar{x}$ is not in Q and there is a separation vector a such that
   $\langle a, x - \bar{x} \rangle \leq \gamma$, $\gamma \leq 0$, $\forall x \in Q \cap Q_0$.

Answers of the first type are named *optimality* cuts, whereas answers of the second type are *feasibility* cuts.

The successive answers of the oracle to a sequence of query points $x^k \in Q_0$ define an outer polyhedral approximation of the problem. Let $\{x^j\}$, $j \in K = \{1, \cdots k\}$, be the sequence of query points. The set K is partitioned into $K = K_1 \cup K_2$, where K_1 and K_2 correspond to optimality cuts and feasibility cuts respectively. Let the oracle answer at x^j be (d^j, c_j), with $(d^j, c_j) = (\xi^j, f(x^j))$, for $j \in K_1$, and $(d^j, c_j) = (a^j, \gamma^j)$ for $k \in K_2$. Let $\theta^k = \min_{j \in K_1}\{f(x^j)\}$ be the best recorded function value. The polyhedral approximation defines, in the epigraph space (x, z), the *set of localization*:

$$\begin{aligned}\mathcal{F}_k = \{(x, z) \mid x \in Q_0,\ z \leq \theta^k,\ & \\ & \langle \xi^j, x - x^j \rangle - z \leq -f(x^j),\ j \in K_1, \\ & \langle a^j, x - x^j \rangle \leq \gamma^j,\ j \in K_2 \}\,.\end{aligned}$$

Clearly, $\mathcal{F}_k$ contains the optimal solutions of the initial problem.

The k-th step of the generic cutting plane algorithm is as follows.

1. Pick $x^k \in \mathcal{F}_k$.
2. The oracle returns the generic cut $\langle d^k, x \rangle \leq c_k$: either a feasibility cut or an optimality cut.
3. Update $\mathcal{F}_{k+1} := \mathcal{F}_k \cap \left\{x \mid \langle d^k, x \rangle \leq c_k\right\}$.

2.4 Interior point based decomposition

2.4.1 Selecting query points

Specific cutting plane algorithms differ in the choice of the query point x^k. Let us focus on two strategies.

The most popular approach in decomposition is to query the oracle at the optimum of the current polyhedral relaxation, i.e., $(\bar{x}, \bar{z}) = \arg\min\{z \mid (x, z) \in \mathcal{F}_k\}$. This corresponds to solving the restricted master program to optimality. This strategy was advocated by Kelley [26] and Cheney-Goldstein [10] in the context of general cutting planes and is the same as in Dantzig-Wolfe [12] or Benders [2]. This strategy selects extreme points in the localization set (basic solutions in the restricted master). This choice often works well, but sometimes turns to a failure: the oracle queries at uninformative points and the method stalls.

To overcome the instability of Kelley's method, many alternative schemes have been proposed. Among them, center methods use some kind of center of the localization set. A choice that turned out to be efficient is to use the analytic center. It was first suggested by Sonnevend [30] in the context of convex optimization. The method was first implemented and tested in [19]. It has been applied in many different contexts and has been proved to be efficient and stable. (For a short survey, see [20]).

To define the analytic center strategy let us introduce the slacks

$$s_j = \begin{cases} z - f(x^j) - \langle \xi^j, x - x^j \rangle,\ j \in K_1, \\ \\ \gamma^j - \langle a^j, x - x^j \rangle,\ j \in K_2, \end{cases}$$

and the associated potential

$$\phi_k(x,z) = F_0(x) - \log(\theta^k - z) - \sum_{j \in K} \log s_j,$$

where $F_0(x)$ is the barrier function associated with the set Q_0. The analytic center strategy selects

$$(x,z) = \arg\min \phi_k(x,z).$$

For a concise summary of the method and its convergence properties, see [20].

In practice, it is often the case that the function f is additive and the set Q is the intersection of many sets. As shown in Section 2.2, this is the case of function (8) which turns out to be the sum of several functions L_i (each one of them corresponding to a subproblem) and the feasible set Q is the intersection of the domains of these functions. In such a case, the oracle returns separate information for each function L_i.

Let us reformulate Problem (10) as

$$\min\{\sum_{i=1}^{m} f_i(x) \mid x \in Q = Q_0 \cap_{i=1}^{p} Q_i\}. \tag{11}$$

For any $x \in Q_0$ the oracle associated with this formulation provides the answers:

1. $\bar{x}$ is in Q and there are support vectors
   $\xi_i \in \partial f_i(\bar{x})$, i.e., $f_i(x) \geq f_i(\bar{x}) + \langle \xi_i, x - \bar{x} \rangle \; \forall x \in R^n$, $i = 1, \cdots m$.

2. $\bar{x}$ is not in Q and there are separation vectors a_i such that
   $\langle a_i, x - \bar{x} \rangle \leq \gamma_i$, $\gamma_i \leq 0$, $\forall i$ such that $\bar{x} \notin Q_i$, $i = 1, \cdots p$, and $\forall x \in Q \cap Q_0$.

The above oracle can be viewed as a collection of *independent* oracles, one for each function f_i and one for each Q_i. This feature has two beneficial consequences. Firstly, the disaggregate formulation much improves the performance of any cutting plane method. Secondly, and most importantly in our case, disaggregation naturally allows parallel computations.

2.4.2 Solving subproblems

The presence of uncertainty in stochastic optimization problems leads to huge size of the problems to be solved. Even when broken down into pieces (subproblems), the sizes of these smaller blocks still remain considerable. In the particular application we are interested in, for example, each subproblem (9) has from few thousand to few tens of thousand rows and columns. This is clearly a demanding optimization problem and we solve it with the LP code [21], an implementation of the infeasible primal-dual interior point method [33]. To save on the computation time we exploit the optimal solution of one problem when solving subsequent one, i.e. we take advantage of the warm start facility of the interior point method [23].

3 Accessing a distributed system via the modeling language

In this section, we first recall how to extract a block structure from algebraic modeling languages. Second we present the parallel decomposition, realized with two home made products: ACCPM [22] and HOPDM [21], that was hooked to the GAMS modeling language. Finally, we present its implementation on a cluster of 10 LINUX PC's.

3.1 Extracting block structures from an algebraic modeling language

Algebraic modeling languages such as AMPL [15] and GAMS [7] are widely used tools among the modeling community. They enable the modelers to build models using equivalent algebraic notations in a manner similar to the one they would use to write them, on a piece of paper. The only problem with these kinds of tool is that they are not adapted to handle very large models. In fact, problems of memory management represent the main bottleneck for solving large scale optimization models. For this reason optimization problems that are generated from modeling languages rarely go beyond the size of 50,000 constraints. In this section we will explain why these problems of memory management occur.

Algebraic modeling languages work the following way. There is a first step where the algebraic formulation, which is written by the modeler, is translated into a mathematical program. In a second step, the mathematical program is optimized with an appropriate solver (e.g., linear, nonlinear). The constraint matrix of the mathematical program is in most cases a sparse matrix. This is due to the fact that each variable occurs only in very few equations of the model. However, although these matrices are sparse most of the time, the number of non-zero elements increases fast with the size of the matrix. Consequently, a general purpose solver that handles a large model generated in the usual manner by a modeling language will encounter enormous difficulties in allocating sufficient memory to solve it. (Technical information about memory allocation related to our experiments is given in section 4.)

An algebraic modeling language can thus be seen as a black-box. The concept behind can be very appealing for the modeler who is not a computer expert. On the other hand, it has some weakness when problems are very large and present, which is often the case, a decomposable structure. Indeed, we saw in Section 2 that decomposable structures can be exploited by the solution algorithm giving to the solver smaller problems to handle. Moreover, subproblems can be solved independently on different nodes of a distributed system. But the problem is that the modeling language does not enable to produce this kind of structure eventhough it was identified in the algebraic formulation of the model.

To overcome this problem we developed SET [17] that stands for Structure Exploiting Tool. This tool enables the modeler to extract the desired structure from the modeling language providing a few addidtional information. The additional information is expressed in the terms of the model syntax. It is indeed important to respect the logical structure of the modeling language, in order to interact with the decomposition method. There is a great wealth of efficient decomposition methods but unfortunately there is no unified way of accessing them from the modeling language. Only few experiences were limited to implementing the decomposition algorithm directly in the interface of the modeling language as in [9] but this way of doing prevents any possibility to link them with a distributed system. SET provided a general support for the decomposition directly from the modeling language.

The main idea underlying this approach is that it is generally sufficient to permute the rows and columns of the matrix to regain the decomposed structure. We need a minimum additional information to generate these permutations, which usually are not unique since we do not care about the permutation inside the blocks. Our tool is then able to produce a mapping which indicates to which subproblem a given equation or variable belongs. To construct the mapping from the index sets (e.g., time, locations, materials) [16], we require information on

- the number of subproblems;
- the set of rows and columns for each subproblem expressed in terms of the modeling language.

3.2 Accessing parallel decomposition from algebraic modeling language

BALP is an interior point decomposition code [23] that calls for two independent solvers, one to compute the analytic center of the localization set and the other one to compute optimal solutions for the subproblems. It is implemented on a cluster of 10 Pentium Pro PC's [31] linked with Ethernet that allows a 10MB/s transfer rate. The operating system is LINUX and the parallel communication is done with MPI [6].

Figure 1 summarizes the whole process. A compact formulation written with equivalent algebraic notations along with the data are fed to the algebraic modeling language. From this point, all operations are automated. The algebraic modeling language produces the complete mathematical

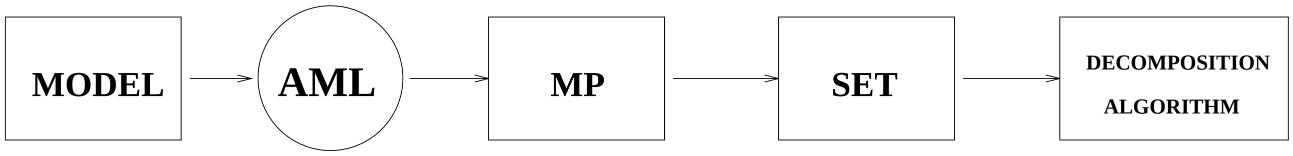

Figure 1: Passing structural information from AML to the solver.

programming formulation. SET provides the structural information on how to split the anonymous matrix produced by the AML into blocks corresponding to the required block-angular formulation. It then dispatches the subproblems to the different PC's.

It is finally left to the decomposition code to retrieve for each subproblem all appropriate elements (i.e. q^l, T^l, W^l, h^l. Given x, as described in Section 2.2, subproblems are solved and return each a vector π^l or a direction d^l.) The interior point solver uses this information to add new constraints to the master problem and compute a new analytic center, etc.

4 Experiments with a financial planning model

In this section, we first describe the formulation of the financial planning model using GAMS notations and the different model sizes generated by GAMS. Then we present the execution times to solve these problems on a cluster of 10 LINUX PC's. We also give some comments on speed-up in a parallel implementation.

4.1 Description of the model

The model considered in this section is a simple financial planning problem suggested in [3] (It is just used here for its generic properties and definitely not for its modeling content). 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 scenario is 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. The corresponding GAMS model can be found in the

Appendix along with its equivalent algebraic formulation.

4.2 Results

Different problem sizes are generated by varying the number of outcomes (or events) from the 6-periods problem. The total number of scenarios is thus given by the formula N^6, with 6 beeing the number of periods and N the number of possible realizations at each time period. We only report in Table 1 experiments where the master contains the first two periods. This choice of cutting the model between the second and the third periods corresponds to the best results we got. For instance, the number of subproblems in $P6R100$ is 100. This problem has 6 periods and 10 realizations per node. Its overall formulation (4) has 1,111,112 constraints and 2,555,556 variables.

Table 1: *Description of problems generated by GAMS.*

Problem	Events	Scenarios	Rows	Columns
P6R9	3	3^6	1094	3279
P6R16	4	4^6	5462	15018
P6R25	5	5^6	19532	50781
P6R36	6	6^6	55988	139968
P6R49	7	7^6	137258	338339
P6R64	8	8^6	299594	711534
P6R81	9	9^6	597872	1395033
P6R100	10	10^6	1111112	2555556

Table 2 shows the number of processors used to solve the different problems. Its last column gives the time to solve these problems with the decomposition algorithm. The $P6R100$ problem is particularly interesting because of its very large size. To our best knowledge, this is the largest model ever generated from a modeling language. It is solved in less than 3 hours. The parallel decomposition was performed on 10 Pentium Pro PCs, each with 200 MHz processor and 64 MB of RAM. To be able to solve the largest models we had to add 384 MB of swap space on each machine. In the decomposition algorithm there is no need to keep all the data in memory at the same time, since while a subproblem is handled by the solver of a specific node, the other suproblems may wait on the swap to be processesed later. This is the main reason why we could solve, for the first time, such a huge model from the modeling language in a short overall execution time. Memory management indeed represents the main bottelneck for solving large optimization problems.

Table 2: *Execution times of the decomposition algorithm.*

Problem	Events	SubProbs	Procs	Time [s]
P6R9	3	9	3	8
P6R16	4	16	4	20
P6R25	5	25	5	49
P6R36	6	36	6	100
P6R49	7	49	7	512
P6R64	8	64	8	1851
P6R81	9	81	9	6656
P6R100	10	100	10	10325

We also analysed the speed-ups obtained by the parallel code. To do this, we focused on a relatively small problem, namely problem $P6R36$ ($P6R64$, $P6R81$ and $P6R100$ could not be solved with fewer nodes). $P6R36$ problem has 55,988 constraints and 139,968 variables. As indicated by its name, it contains 36 subproblems. We thus ran successively this problem on 1 to 10 processors.

As shown in Table 3, the speed-up corresponds to what we could reasonably expect with the decomposition algorithm. These results confirm (once again) that decomposition is particularly well suited to parallel implementation.

5 Conclusion

We showed in this paper that advanced decomposition software and parallel computations can be directly accessed from a modeling language at a minimal cost for the user. The approach takes advantage of the fact that most models contain embedded block structures that can be exploited by parallelisable optimization techniques, e.g., decomposition algorithms. So, in this framework the modeling language can be viewed as the manager of the distributed system. The modeler becomes more involved in the solution process. He/She intelligently exploits his/her personal insight into the model to increase the efficiency of the solution process.

To show the feasibility of this approach we implemented it on a cluster of 10 Pentium Pro PC's. Stochastic financial planning models were generated with GAMS, a widely used algebraic modeling language. A decomposable block structure was then automatically extracted and sub-blocks were

Table 3: *Speed-up for the $P6R36$ problem.*

Processors	Time [s]	Speed-up
1	537.7	1.0
2	275.9	1.95
3	188.0	2.86
4	147.2	3.65
5	126.8	4.24
6	100.2	5.36
7	99.9	5.38
8	87.5	6.14
9	82.4	6.52
10	81.9	6.56

appropriately distributed on the cluster of PC's according to the modeler inputs. Finally, a parallelized interior point decomposition method solves the problem. Decomposition is an algorithmic device that breaks down computations into several independent sub-problems. It is thus ideally suited to distributed computations, and to problems that are too large to be handled with a frontal approach. Our largest problem, a financial planning model with one million scenarios, has 1,111,112 equations and 2,555,556 variables. It was solved using this framework in less than three hours. So, we hope to have contributed to making parallel large scale optimization accessible to the modeling community.

References

[1] O. Bahn, O. du Merle, J.-L. Goffin, and J.-P. Vial, *A cutting plane method from analytic centers for stochastic programming*, Mathematical Programming, 69 (1995), pp. 45–73.

[2] J. F. Benders, *Partitioning procedures for solving mixed-variables programming problems*, Numerische Mathematik, 4 (1962), pp. 238–252.

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

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

[5] J. R. Birge and L. Qi, *Computing block-angular Karmarkar projections with applications to stochastic programming*, Management Science, 34 (1988), pp. 1472–1479.

[6] 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.

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

[8] D. Carino, T. Kent, D. Myers, C. Stacy, M. Sylvanus, A. Turner, K. Watanabe, and W. Ziemba, *The Russel-Yasuda Kasai model: an asset/liability model for Japanese insurance company using multistage stochastic programming*, Interface, 24 (1994), pp. 29–49.

[9] D. Chang, *Solving Dynamic Equilibrium Models with Multistage Benders Decomposition*, PhD thesis, Stanford University, Department of Engineering Economic Systems and Operations Research, October 1997.

[10] E. Cheney and A. Goldstein, *Newton's method for convex programming and Tchebycheff approximation*, Numerische Mathematik, 1 (1959), pp. 253–268.

[11] C. Condevaux-Lanloy and E. Fragniere, *Setstoch: a tool for multistage stochastic programming with recourse*, tech. rep., Logilab, University of Geneva, 102 Bd Carl-Vogt, CH-1211, June 1998 (in preparation).

[12] G. B. Dantzig and P. Wolfe, *The decomposition algorithm for linear programming*, Econometrica, 29 (1961), pp. 767–778.

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

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

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

[16] E. Fragnière, J. Gondzio, and R. Sarkissian, *Customized block structures in algebraic modeling languages: the stochastic programming case*, in Proceedings of the CEFES/IFAC98, S. Holly, ed., Amsterdam, 1998, Elsevier Science.

[17] E. Fragnière, 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.

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

[19] J.-L. Goffin, A. Haurie, and J.-P. Vial, *Decomposition and nondifferentiable optimization with the projective algorithm*, Management Science, 38 (1992), pp. 284–302.

[20] J.-L. Goffin and J.-P. Vial, *Interior point methods for nondifferentiable optimization*, in Operations Research Proceedings 1997, P. Kischka, H.-W. Lorenz, U. Derigs, W. Domschke, P. Kleinschmidt, and R. Möhring, eds., Springer Verlag, Berlin, Germany, 1998, pp. 35–49.

[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 and J.-P. Vial, *Warm start and ε-subgradients in cutting plane scheme for block-angular linear programs*, tech. rep., Logilab, University of Geneva, 102 Bd Carl-Vogt, CH-1211, June 1997, revised in November 1997. To appear in *Computational Optimization and Applications.*

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

[25] E. R. Jessup, D. Yang, and S. A. Zenios, *Parallel factorization of structured matrices arising in stochastic programming*, SIAM Journal on Optimization, 4 (1994), pp. 833–846.

[26] J. E. Kelley, *The cutting plane method for solving convex programs*, Journal of the SIAM, 8 (1960), pp. 703–712.

[27] R. C. Leachman, Roberti, F. Benson, and C. Lui, *Impress: An automated production-planning and delivery-quotation system at harris corporation-semiconductor sector*, Interfaces, 26 (1996), pp. 6–37.

[28] J. Mulvey and A. Ruszczyński, *A new scenario decomposition method for large scale stochastic optimization*, Operations Research, 43 (1995), pp. 477–490.

[29] A. Ruszczyński, *A regularized decomposition method for minimizing a sum of polyhedral functions*, Mathematical Programming, 33 (1985), pp. 309–333.

[30] G. Sonnevend, *New algorithms in convex programming based on a notion of "centre" (for systems of analytic inequalities) and on rational extrapolation*, in Trends in Mathematical Optimization : Proceedings of the 4th French–German Conference on Optimization in Irsee, Germany, April 1986, K. H. Hoffmann, J. B. Hiriat-Urruty, C. Lemarechal, and J. Zowe, eds., vol. 84 of International Series of Numerical Mathematics, Birkhäuser Verlag, Basel, Switzerland, 1988, pp. 311–327.

[31] 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.

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

[33] S. Wright, *Primal-Dual Interior-Point Methods*, SIAM, Philadelphia, 1997.

[34] S. Zenios, *Parallel and supercomputing in the practice of management science*, Interface, 24 (1994), pp. 122–140.

[35] ——, *High-performance computing and networking*, Private communication, (June 9, 1998).

A Multistage Financial Model

Indexes:

i: asset,

t: stage or period,

l_t: node number of stage t,

L_t: last node number of stage t,

$a(l_t)$: immediate ancestor of node l_t.

Variables:

C^{l_t}: cash (in tausend Dollars),

$X_i^{l_t}$: asset (in tausend Dollars),

U^{l_T}: surplus end of period (in tausend Dollars),

V^{l_T}: deficit end of period (in tausend Dollars).

Parameters:

w_i: initial capital (in tausend Dollars),

w_f: final capital (in tausend Dollars),

r_c: return of cash,

$p_i^{l_t}$: price of assets.

$$
\begin{array}{llll}
\max & \displaystyle\sum_{l_T=L_{T-1}+1}^{L_T} 5U^{l_T} - 20V^{l_T} & & \\
\text{s.c.} & & & \\
w_i & = & C^{l_0} + \displaystyle\sum_{i=1}^{n} X_i^{l_0}, & l_0 = 1, \\
r_c C^{l_0} + \displaystyle\sum_{i=1}^{n} r_i^{l_1} X_i^{l_0} & = & C^{l_1} + \displaystyle\sum_{i=1}^{n} X_i^{l_1}, & l_1 = 2, \ldots, L_1, \\
r_c C^{a(l_2)} + \displaystyle\sum_{i=1}^{n} p_i^{l_2} X_i^{a(l_2)} & = & C^{l_2} + \displaystyle\sum_{i=1}^{n} X_i^{l_2}, & l_2 = L_1 + 1, \ldots, L_2, \\
\vdots & & \vdots & \vdots \\
r_c C^{a(l_T)} + \displaystyle\sum_{i=1}^{n} p_i^{l_T} X_i^{a(l_T)} & = & w_f + U^{l_T} - V^{l_T}, & l_T = L_{T-1} + 1, \ldots, L_T, \\
C^{l_t}, \quad X_i^{l_t}, \quad U^{l_T}, \quad V^{l_T} \geq 0, & & i = 1, \ldots, n, & t = 1, \ldots, T, \quad l_t = L_{t-1} + 1, \ldots, L_t.
\end{array}
$$

B GAMS model

This section resproduces one of the models that was generated by GAMS and solved on the cluster of PC's. This specific one creates a linear program with equations and variables. To keep a very compact formulation we kept the same sample of prices for every periods. Those who are familiar with GAMS know that it is not a tool intended to produce multistage stochastic programs (see [11] for more detail on how to access a stochastic program generator form algebraic modeling languages). Here, we are using the ALIAS command to generate symetric event tree distribution.

```
* Four-Period Portfolio

OPTION RESLIM = 10000;
OPTION ITERLIM = 25000;
OPTION LIMROW = 0;
OPTION LIMCOL = 0;
OPTION SOLPRINT = OFF;

SETS
  S Scenarios    /S1, S2, S3, S4, S5, S6, S7, S8, S9, SA0/
  A  Assets      /USAB, FORS, CORP,  GOVE/;

ALIAS (S,O,D,I,R,E);

SCALARS
  RC  Cash rate of return    /1.05/
  WI  Initial capital        /50/
  WF  Goal                   /75/;

TABLE P(S,A)  Asset rates of return
      USAB   FORS   CORP   GOVE
S1    1.27   1.16   0.99   1.02
S2    1.20   1.41   1.04   1.05
S3    1.06   0.91   1.11   1.10
S4    1.23   0.83   1.05   1.04
S5    1.09   1.10   0.95   0.98
S6    1.15   1.28   1.25   1.18
S7    0.83   0.97   1.02   1.07
S8    0.83   0.77   0.91   0.98
S9    1.09   0.96   1.03   1.03
SA0   1.20   1.18   1.18   1.16;
```

```
VARIABLES
  EU
POSITIVE VARIABLES
  C                   Cash Period 0
  C1(S)               Cash Period 1
  C2(S,O)             Cash Period 2
  C3(S,O,D)           Cash Period 3
  C4(S,O,D,I)         Cash Period 4
  C5(S,O,D,I,R)       Cash Period 5
  X(A)                Asset Period 0
  Y(S,A)              Asset Period 1
  Z(S,O,A)            Asset Period 2
  W(S,O,D,A)          Asset Period 3
  N(S,O,D,I,A)        Asset Period 4
  M(S,O,D,I,R,A)      Asset Period 5
  U(S,O,D,I,R,E)      Surplus Period 6
  V(S,O,D,I,R,E)      Deficit Period 6;

EQUATIONS
  OBJECTIVE       Calculating the expectation of the utility function
  G                   Balance of Financial Flows Period 0
  H(S)                Balance of Financial Flows Period 1
  T(S,O)              Balance of Financial Flows Period 2
  F(S,O,D)            Balance of Financial Flows Period 3
  Q(S,O,D,I)          Balance of Financial Flows Period 4
  B(S,O,D,I,R)        Balance of Financial Flows Period 5
  L(S,O,D,I,R,E)      Balance of Financial Flows Period 6;


OBJECTIVE..       EU=E=SUM((S,O,D,I,R,E),5*U(S,O,D,I,R,E)-20*V(S,O,D,I,R,E));
G..               SUM(A,X(A))+C =L= WI;
H(S)..            SUM(A,X(A)*P(S,A))+C*RC =G= SUM(A,Y(S,A))+C1(S);
T(S,O)..          SUM(A,Y(S,A)*P(O,A))+C1(S)*RC =E= SUM(A,Z(S,O,A))+C2(S,O);
F(S,O,D)..        SUM(A,Z(S,O,A)*P(D,A))+C2(S,O)*RC =E= SUM(A,W(S,O,D,A))+C3(S,O,D);
Q(S,O,D,I)..      SUM(A,W(S,O,D,A)*P(I,A))+C3(S,O,D)*RC =E=
                  SUM(A,N(S,O,D,I,A))+C4(S,O,D,I);
B(S,O,D,I,R)..    SUM(A,N(S,O,D,I,A)*P(R,A))+C4(S,O,D,I)*RC =E=
                  SUM(A,M(S,O,D,I,R,A))+C5(S,O,D,I,R);
L(S,O,D,I,R,E).. SUM(A,M(S,O,D,I,R,A)*P(E,A))+C5(S,O,D,I,R)*RC =E=
                  WF+U(S,O,D,I,R,E)-V(S,O,D,I,R,E);


MODEL PORT /ALL/;


OPTION LP = ZOOM;

SOLVE PORT USING LP MAXIMAZING EU;

DISPLAY X.L, C.L;
```