# Parallelization and Aggregation of Nested Benders Decomposition

**M.A.H. Dempster and R.T. Thompson**
**Department of Mathematics**
**University of Essex**
**Wivenhoe Park**
**Colchester**
**England CO4 3SQ**
**mahd@essex.ac.uk and thomro@essex.ac.uk**

## Abstract

Dynamic multistage stochastic linear programming has many practical applications for problems whose current decisions have to be made under future uncertainty. There are a variety of methods for solving these problems, including nested Benders decomposition. In this method, recently shown to be superior to the alternatives for large problems, the problem is decomposed into a set of smaller linear programming problems. These problems can be visualised as being attached to the nodes of a tree which is formed from the realizations of the random data vectors determining the uncertainty in the problem. The tree is traversed forwards and backwards, with information from the solutions to each nodal linear programming problem being passed to its immediate descendants by the formation of their right hand sides and to its immediate ancestor in the form of cuts. Problems in the same time period can be solved independently and it is this inherent parallelism that is exploited in our parallel nested Benders algorithm. A parallel version of the MSLiP nested Benders code has been developed and tested on various types of MIMD machines. The differing structures of the test problems cause differing levels of speed-up. Results show that problems with few variables and constraints per node do not gain from this parallelization. Stage aggregation has been successfully explored for such problems to improve their parallel solution efficiency by increasing the size of the nodes and therefore the time spent calculating relative to the time spent communicating between processors.

**Key Words:** Linear programming, dynamic stochastic programming, nested Benders decomposition, parallel algorithms, aggregation, MIMD computers

## 1 Introduction

Stochastic programs have many practical applications for problems where a decision has to be made now, but where the effectiveness of the decision is dependent on future uncertainty. In these models the future is revealed in time stages $t$. For practical purposes it is assumed that the number of stages is finite and the problem data sample space $\Omega$ has a finite number of elements. A filtration $\mathcal{F} := \{\mathcal{F}_t : t = 1, \ldots, T\}$ is defined on $\Omega$, where each $\sigma$–algebra $\mathcal{F}_t$ consists of the data events which have been revealed by period $t$. A random vector $\boldsymbol{\omega}$ can be defined on the sample space as the trivial map: $\omega \to \omega$, $\omega \in \Omega$. The sample space and filtration, combined with a probability measure, define a filtered probability space for the model data.

Within this framework a variety of models have been developed. For an overview of stochastic programming see [18, 19, 25, 31, 22]. One such model is the *dynamic* or *multistage stochastic linear programme with recourse.* This model has been applied to natural resource management [51, 45, 41], portfolio management [10, 29, 35, 20, 21, 39, 12, 13] , and resource acquisition [6]. The multistage stochastic linear programming model can be written as

$$\min_{x_1}\{c_1' x_1 + \mathbb{E}_{\boldsymbol{\omega}_2}\{\min_{\mathbf{x}_2} \mathbf{c}'_2\mathbf{x}_2 + \mathbb{E}_{\boldsymbol{\omega}_3|\boldsymbol{\omega}_2}\{\min_{\mathbf{x}_3} \mathbf{c}'_3\mathbf{x}_3 + \ldots + \mathbb{E}_{\boldsymbol{\omega}_T|\boldsymbol{\omega}_2,\ldots,\boldsymbol{\omega}_{T-1}}\{\min_{\mathbf{x}_T} \mathbf{c}_T\mathbf{x}_T\}\}$$

$$\begin{array}{lllllll}
\text{s.t. } A_1x_1 & & & & & = & b_1 \\
\mathbf{B}_2x_1 & + & \mathbf{A}_2\mathbf{x}_2 & & & = & \mathbf{b}_2 \; a.s \\
 & & \mathbf{B}_3\mathbf{x}_2 & + & \mathbf{A}_3\mathbf{x}_3 & = & \mathbf{b}_3 \; a.s. \\
 & & & \ddots & & & \vdots \\
 & & & \mathbf{B}_T\mathbf{x}_{T-1} & + \; \mathbf{A}_T\mathbf{x}_T & = & \mathbf{b}_T \; a.s.
\end{array} \tag{1}$$

$$\begin{array}{lllll}
l_1 & \leq & x_1 & \leq & u_1 \\
\mathbf{l}_t & \leq & \mathbf{x}_t & \leq & \mathbf{u}_t \quad t = 1, .., T,
\end{array}$$

where $\boldsymbol{\omega}_t = (\boldsymbol{b}_t, \boldsymbol{c}_t, \boldsymbol{A}_{t,1}, \ldots, \boldsymbol{A}_{t,n_t}, \boldsymbol{B}_{t,1}, \ldots, \boldsymbol{B}_{t,n_t})$, $t = 2, ..., T$, are random vectors in some canonical probability space $(\Omega, \mathcal{F}, \mathcal{P})$. The subindexes of the matrices B, A, and the vectors c, x ,b refer to the stage of the problem. Once the realized values are observed at a stage t, the information required to decide the actions at stage t+1 is known.

The size of the problem increases rapidly with addition of both scenarios, i.e. data paths $\omega_t$, and stages $t = 2, .., T$. Generally this model is solved by taking the *deterministic equivalent problem*, either the relatively compact standard form or the extended split variable form. In the *split variable form* the *non–anticipativity* conditions, expressing the fact that the decisions must be based only on current and past information, are written out explicitly.

There are a variety of well–known algorithms for solving the deterministic equivalent problem: the *simplex method*, [37, 30, 14, 15, 23, 46], *interior point* methods [30, 14, 38, 49] *augmented Lagrangian* [5, 19, 44] and *scenario decomposition methods* [44, 3] and *nested Benders decomposition.* Benders decomposition [4] is the dual form of the algorithm introduced by Dantzig and Wolfe [17]. The dual form is more appropriate for the matrix structure presented by the deterministic form of the two–stage stochastic linear programming problem. This method is a cutting plane algorithm [33] and was proposed by Van Slyke and Wets [50] and developed into an effective algorithm by Kallberg and Ziemba [32]. The algorithm was extended to the multistage case by Birge, using Benders recursively to give the *nested Benders* algorithm. This has been developed into code by Gassmann [27](MSLiP), Birge *et al* [9](ND–UM) and King [34](SP/OSL). Improvements to MSLiP have been made by Thompson [47] with the replacement of the Pfefferkorn–Tomlin LP code [42] with OSL's LP solver (MSLiP–OSL).

Recently interest in the development of parallel algorithms for solving the multistage stochastic linear programming problem has increased. Various parallelization methods have been applied to the algorithms mentioned earlier. The interior point method can be parallelized efficiently by vectorizing the Cholesky factorization [36, 53]. Scenario decomposition methods have a natural parallel decomposition structure [40, 43, 39]. Parallel versions of Benders decomposition have also been developed [2, 24, 8].

In Section 2 we present the nested Benders algorithm, its implementation in the MSLiP code and improvements to the default solution time which can be made through sequencing and cut formation. We introduce aggregation as a method of changing the structure of the problem by increasing the size of subproblems and reducing the number of stages in Section3. We present results for a set of test problems, discussing the problem sets, difficulties with obtaining solutions, solution times, improvements to solution times through aggregation and the stochasticity of the problems solved. The parallel implementation of the MSLiP code is described in Section 4 and computational results are reported for the problem sets on Fujitsu's AP1000 computer [26] . Improvements to the parallelization made by stage aggregation are also discussed. Section 5 contains the conclusions and suggests areas for further development.

## 2 Nested Benders decomposition

Traditionally, multistage stochastic linear programming problems have been solved by taking the *deterministic equivalent* form

$$\min\left\{c_1x_1+\sum_{k_2=1}^{K_2}p_2^{k_2}c_2^{k_2}x_2^{k_2}+\sum_{k_2=1}^{K_2}\sum_{k_3=1}^{K_3}p_2^{k_2}p_3^{k_3}c_3^{k_3}x_3^{k_2,k_3}+\ldots+\sum_{k_2=1}^{K_2}\cdots\sum_{k_T=1}^{K_T}p_2^{k_2}\ldots p_T^{k_T}c_T^{k_T}x_T^{k_2,\ldots,k_T}\right\}$$

$$\begin{array}{llllllll}
\text{s.t.} & A_1x_1 & & & & & = & b_1\\
& B_2^{k_2}x_1 & + & A_2^{k_2}x_3^{k_2} & & & = & b_2^{k_2}\\
& & & B_3^{k_2,k_3}x_2^{k_2} & + & A_3^{k_2,k_3}x_3^{k_2,k_3} & = & b_1^{k_2,k_3}\\
& & & & \ddots & \ddots & & \vdots\\
& & & & & B_T^{k_2,\ldots,k_{T-1}}x_{T-1}^{k_2,\ldots,k_{T-1}} \;+\; A_T^{k_2,\ldots,k_T}x_T^{k_2,\ldots,k_T} & = & b_T^{k_2,\ldots,k_T}
\end{array}\qquad(2)$$

$$\begin{array}{lllll} l_1 & \leq & x_1 & \leq & u_1 \\ l_t^{k_2,\ldots,k_t} & \leq & x_t^{k_2,\ldots,k_t} & \leq & u_t^{k_2,\ldots,k_t} \\ & & k_t = 1,\ldots,K_t & & t = 2,\ldots,T, \end{array}$$

in which constraints are written out explicitly for each possible realization of the data process. In this form the problem can be solved by the simplex method, and more recently by interior point methods.

The nested Benders method solves problem (2) in a recursive manner. This can be illustrated by considering a particular node of the decision tree and its descendants. Taking node $n$ at stage $t$ with *parent* $\alpha(n)$ and *descendants* $\omega(n)$ we can write the corresponding subproblem as

$$\begin{array}{lll} & \min\{c_t^n x_t^n + Q(x_t^n) & : \quad A_t^n x_t^n = b_t^n - B_t^{\alpha(n)} x_{t-1}^{\alpha(n)}\} \\ = & \min\{c_t^n x_t^n + \theta_t^n & : \quad A_t^n x_t^n = b_t^n - B_t^{\alpha(n)} x_{t-1}^{\alpha(n)},\ \theta_t^n \geq Q(x_t^n)\} \end{array} \tag{3}$$

where

$$\begin{array}{lll} Q(x_t^n) & := & \min \sum_{\omega(n)=1}^{K_t+1} p^{\omega(n)} c_{t+1}^n x_{t+1}^{\omega(n)} \\ & & \\ \text{s.t.} & & A_{t+1} x_{t+1}^{\omega(n)} = b_{t+1}^{\omega(n)} - B_{t+1}^{\omega(n)} x_t^n \\ & & l_t^n \leq x_t^n \leq u_t^n, \\ & & l_{t+1}^{\omega(n)} \leq x_{t+1}^{\omega(n)} \leq u_{t+1}^{\omega(n)}, \quad \omega(n) = 1,..,K_{t+1}. \end{array} \tag{4}$$

For any value $\tilde{x}_t^n$ we can solve the problem $Q(\tilde{x}_t^n)$ or its *dual equivalent*

$$\begin{array}{ll} \max \sum_{\omega(n)=1}^{K_{t+1}} & p^{\omega(n)}\{(b_{t+1}^{\omega(n)} - B_{t+1}^{\omega(n)} \tilde{x}_t^n)\pi_{\omega(n)} + l_{t+1}^{\omega(n)} \lambda_{\omega(n)} - u_{t+1}^{\omega(n)} \mu_{\omega(n)}\} \\ & \\ \text{s.t.} & \lambda_{\omega(n)} - \mu_{\omega(n)} + A_{t+1}^{\omega(n)} \pi_{\omega(n)} = c_{t+1}^{\omega(n)} \\ & \lambda_{\omega(n)} \geq 0, \quad \mu_{\omega(n)} \geq 0 \quad \omega(n) = 1,\ldots,K_{t+1}. \end{array} \tag{5}$$

Both the primal and dual problems can be decomposed into $K_{t+1}$ smaller subproblems. The dual problems are all feasible since $\pi_{\omega(n)}$ is unconstrained.

If the dual problem (5) is unbounded, then the primal problem is infeasible, and thus there exists $(\sigma^*, \lambda^*, \mu^*)$ such that

$$\sigma^*(b_{t+1}^{\tilde{\omega}(n)} - B_{t+1}^{\tilde{\omega}(n)} \tilde{x}_t^n) + \lambda^* l_{t+1}^{\tilde{\omega}(n)} - \mu^* u_{t+1}^{\tilde{\omega}(n)} \ > \ 0.$$

To try to induce feasibility in the primal, we add the constraint

$$\sigma^*(b_{t+1}^{\tilde{\omega}(n)} - B_{t+1}^{\tilde{\omega}(n)} \tilde{x}_t^n) + \lambda^* l_{t+1}^{\tilde{\omega}(n)} - \mu^* u_{t+1}^{\tilde{\omega}(n)} \ \leq \ 0, \tag{6}$$

termed a *feasibility cut.*

Otherwise, let $(\pi_{\omega(n)}^*, \lambda_{\omega(n)}^*, \mu_{\omega(n)}^*)$, $\omega(n) = 1,...,K_{t+1}$, be an optimal solution for (5) and, by convexity of Q, we have the following inequality

$$Q(x_t^n) \ \geq \ \sum_{\omega(n)=1}^{K_{t+1}} p^{\omega(n)}[\pi_{\omega(n)}^*(b_{t+1}^{\omega(n)} - B_{t+1}^{\omega(n)} x_t^n) + \lambda^* l_{t+1}^{\omega(n)} - \mu^* u_{t+1}^{\omega(n)}] \quad \forall\, x_t^n, \tag{7}$$

where equality holds for $\tilde{x}_t^n$. This produces one extra constraint in the parent problem, a single *optimality cut.*

Then having a collection of dual feasible vectors $(\pi_1^i, \lambda_1^i, \mu_1^i, \ldots, \pi_{K_{t+1}}^i, \lambda_{K_{t+1}}^i, \mu_{K_{t+1}}^i)$, $i = 1, ..., I$, and directions $(\sigma_{\omega(n(j))}^j, \lambda_{\omega(n(j))}^j, \mu_{\omega(n(j))}^j)$, $j = 1, ..., J$, we obtain the problem

$$\begin{array}{lll}
\min & c_t^n x_t^n + \theta_t^n & \\
\text{s.t.} & A_t^n x_t^n & = b_t^n x_t^n - B_t^{\alpha(n)} x_{t-1}^{\alpha(n)} \\
& \sum_{\omega(n)=1}^{K_{t+1}} p^{\omega(n)} \pi_{\omega(n)}^i B_{t+1}^{\omega(n)} x_t^n + \theta_t^n & \geq \sum_{\omega(n)=1}^{K_{t+1}} p^{\omega(n)} [\pi_{\omega(n)}^i b_{t+1}^{\omega(n)} + \lambda_{\omega(n)}^i l_{t+1}^{\omega(n)} - \mu_{\omega(n)}^i u_{t+1}^{\omega(n)}] \\
& \sigma_{\omega(n(j))}^j B_{t+1}^{\omega(n(j))} x_t^n & \geq \sigma_{\omega(n(j))}^j b_{t+1}^{\omega(n(j))} + \lambda_{\omega(n(j))}^j l_{t+1}^{\omega(n)} - \mu_{\omega(n(j))}^j u_{t+1}^{\omega(n)} \\
& l_t & \leq x_t \quad \leq u_t \\
& l_t^{\omega(n)} & \leq x_t^{\omega(n)} \leq u_t^{\omega(n)}
\end{array}$$

$$i = 1, ..., I, \;\; j = 1, ..., J, \;\; \omega(n) = 1, ..., K_{t+1}. \tag{8}$$

If $\theta_t^n \geq Q(x_t^{n*})$, the problem is solved . The relaxed problem is solved iteratively, producing optimality and feasibility cuts until the solution to the original problem is obtained.

There is an alternative method of producing optimality cuts. In the case of the *single cut* (7) weighted sums are taken. These can be disaggregated to give a set of constraints in (8) of the form

$$\begin{array}{ccc}
\pi_{\omega(n(i))}^i B_{t+1}^{\omega(n(i))} x_t^n + \theta_t^{\omega(n(i))} & \geq & \pi_{\omega(n(i))}^i b_{t+1}^{\omega(n(i))} + \lambda_{\omega(n(i))}^i l_{t+1}^{\omega(n(i))} - \mu_{\omega(n(i))}^i u_{t+1}^{\omega(n(i))} \\
i = 1, ..., I & &
\end{array} \tag{9}$$

termed *multicuts.*

It can be seen that the node $n$ passes primal information to its children in the formation of their right hand sides and the children pass dual information to their parent in the form of cuts. After solving the node $n$, nested Benders decomposition decides whether to move further down the tree or back up the tree. If a node is found to be infeasible, there is no primal value to pass down the tree, and so information can only be passed back up the tree.

The *nested Benders decomposition* algorithm can be stated as follows.

**Step 0**. Solve the root node (first stage problem)
Set $\theta_t^n := -\infty \; \forall n, \forall t$ and set $t := 1, n := 1$ Solve the *master* problem

$$\min c_1 x_1 \;\; \text{s.t} \;\; A_1 x_1 = b_1.$$

If it is infeasible, STOP. The problem is infeasible.
Otherwise set t=t+1.

**Step 1**. Solve the appropriate nodes in the current time stage $t$.
Some nodes in this time stage will not be solved due to the infeasibility of their parent or

parent's siblings. If a node is found to be infeasible, place a feasibility cut (6) in the parent node. This branch of the tree is not explored any further until all the other nodes are re–solved to optimality.

**Step 2**. Place cuts and move to the previous time stage, or form new right hand sides and move to the next time stage. For nodes which are feasible, the primal values of the subproblems can be passed to their descendants to form their right hand sides, or single (7) or multiple (9) optimality cuts can be placed in the parent node problem. If the current stage is $T$, the horizon, optimality cuts must be placed in the parent node problems. The order in which the tree is traversed is decided by the *sequencing protocol.* Depending on the sequencing protocol and feasiblity of nodes, either set $t := t + 1$ or $t := t - 1$ . If there are no new cuts or right hand sides produced, **go to 3**.
**Go to 1**.

**Step 3**. The problem is solved. The optimal value is given by the root node. This node's primal decisions give the present decisions to be taken, while nodes at other time stages give the decisions to be taken after the realizations of the stochastic data process up to that time stage.

There are several implementations of nested Benders decomposition: MSLiP [28], ND–UM [9], and SP/OSL [34]. We have been working with variants of the MSLiP FORTRAN code.

## MSLiP

The MSLiP code consists of the following modules:

1. An LP solver. The LP solver found in the code is a modification of the Pfefferkorn–Tomlin LP code [42]. It uses the product form of inverse, which tends to be relatively slow compared with the more modern factorizations based on permuting either the lower or upper Hessenberg matrix ( such as the Forest–Tomlin and Bartels–Golub factorizations). The pricing has recently been improved to include steepest edge and random pricing, as well as the default most negative cost. It has no crash–to–feasibilty heuristics.

2. The 'brain' of the code decides whether and how to place cuts at the previous time stage, and whether to move forward or back in the tree.

3. The cut routines form the optimality or feasibility cuts to be placed in the appropriate nodal problem.

4. The trickling routine tries to reduce the amount of calculation done in solving a set of LP problems which differ only in their right hand sides. These problems may share

bases and pivots, and may even have the same optimal basis, see [27].

5. After the solution has been reached there is a routine for calculating the *expected value of perfect information (EVPI)*. This is a useful indicator in determining how 'stochastic' the problem actually is. It has recently been used in the formulation of an extremely effective sampling algorithm [11], which can reduce the size of the problem to be solved dramatically and hence either save computational time or make tractable previously intractable problems.

The sequencing protocol is very important in minimizing the number of LP subproblems to be solved to obtain the solution of the original problem. Wittrock's *fast–forward–fast–back* method [52] developed for deterministic dynamic problems appears to be the most efficient way of traversing the tree. This method states that once stage t is solved either:

1. explore stage t-1, if the current direction is backwards,
2. or explore stage t+1, if the current direction is forwards.

A particular problem is only re–solved when new information is available to it. If a subproblem is infeasible when single cuts are being placed, the subproblems which have the same parent are not solved. Other methods of traversing the tree are *fast–forward* and *fast–back* (see [9, 27]).

Further details of MSLiP can be found in Gassmann [27, 28] and Birge *et al* [9]. The main weakness with the MSLiP code is its LP solver. This has serious problems with large nodal subproblems, especially when many cuts are produced. The ND–UM code, a hybrid code MSLiP–OSL [47] and the SP/OSL code [34] have all overcome this difficulty by using IBM's OSL simplex code. The ND–UM code has the added bonus of being written in C and can use C's dynamic memory allocation to save space. The crash and presolve facilities of MSLIP–OSL can be switched on for problems where there are large numbers of cuts, some of which may be nearly parallel. Other hybrid codes [14, 23] formed with MSLiP were found to be less stable and less accurate than MSLiP–OSL.

## 3 Problem aggregation

An additional routine has been added to the MSLiP and MSLiP–OSL codes, which allows stages to be aggregated.

**Proposition 3.1** *Converting a three–stage stochastic linear programming problem to a two–stage stochastic linear programing problem by either combining the first two stages or by combining the last two stages, proves that aggregation can be applied to any multistage stochastic LP problem using recursion.*

**Proof** Any number of stages can be combined by combining two stages at a time recursively. Stages can be combined with the previous stage or with the next stage. Therefore we can aggregate any number of stages in any manner. ■

The deterministic equivalent LP formulation of the three–stage stochastic linear problem is

$$\min\left\{c_1x_1+\sum_{k_2=1}^{K_2}p_2^{k_2}c_2^{k_2}x_2^{k_2}+\sum_{k_2=1}^{K_2}\sum_{k_3=1}^{K_3}p_2^{k_2}p_3^{k_3}c_3^{k_2,k_3}x_3^{k_2,k_3}\right\}$$

$$\begin{array}{llllll}\text{s.t. } A_1x_1 & & & & = & b_1\\ B_2^{k_2}x_1 & + & A_2^{k_2}x_2^{k_2} & & = & b_2^{k_2}\\ & & & & & k_2=1,\dots,K_2\\ & & B_3^{k_2,k_3}x_2^{k_2} & + \quad A_3^{k_2,k_3}x_3^{k_2,k_3} & = & b_2^{k_2,k_3}\\ & & & & & k_3=1,\dots,K_3\end{array} \tag{10}$$

$$\begin{array}{lllll} l_1 & \le & x_1 & \le & u_1\\ l_t & \le & x_t^{k_2,\dots,k_t} & \le & u_t \quad t=2,3.\end{array}$$

The upper indices $k_1,\dots,k_t$, $t=2,3$, refer to a mutually exclusive outcome of the random data variables at stages 2 and 3.

If we take in (10)

$$d_1:=\begin{pmatrix}b_1\\ b_2^1\\ \vdots\\ b_2^{k_2}\\ \vdots\\ b_2^{K_2}\end{pmatrix},y_1:=\begin{pmatrix}x_1\\ x_2^1\\ \vdots\\ x_2^{k_2}\\ \vdots\\ x_2^{K_2}\end{pmatrix},q_1:=\begin{pmatrix}c_1\\ p^1c_2^1\\ \vdots\\ p^{k_2}c_2^{k_2}\\ \vdots\\ p^{K_2}c_2^{K_2}\end{pmatrix} \tag{11}$$

$$C_1:=\begin{pmatrix}A_1 & & & & \\ B_2^1 & A_2^1 & & & \\ \vdots & & \ddots & & \\ B_2^{k_2} & & & A_2^{k_2} & & \\ \vdots & & & & \ddots & \\ B_2^{K_2} & & & & & A_2^{K_2}\end{pmatrix} D_3^{k_2,k_3}:=\left(\ \dots\ \dots\ \overset{(k_2+1^{th}\ \text{col})}{B_2^{k_2,k_3}}\dots\quad\dots\ \dots\right), \tag{12}$$

then we can rewrite our problem as

$$\begin{array}{llll}\min_{y_1}\{q_1y_1+\mathbb{E}_{\omega_3|\omega_2}\{\min_{\mathbf{x}_3}\mathbf{c}_3\mathbf{x}_3\} & & & \\ s.t.\ C_1y_1 & & = & d_1\\ \mathbf{D}_3\mathbf{y}_1 & + \quad \mathbf{A}_3\mathbf{x}_3 & = & \mathbf{b_3}\quad a.s..\end{array} \tag{13}$$

Therefore a three–stage problem can be converted to a two–stage problem by combining the first two–stages.

However, if we take

$$F_2^{k_2} := \begin{pmatrix} B_2^{k_2} \\ 0 \\ \vdots \\ \vdots \\ \vdots \end{pmatrix} \left.\vphantom{\begin{pmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \end{pmatrix}}\right\} k_3 \text{ zero matrices}, \qquad \mathbb{E}_2^{k_2} := \begin{pmatrix} A_2^{k_2} & & & \\ B_3^{k_2,1} & A_3^{k_2,1} & & \\ \vdots & & \ddots & & \\ B_3^{k_2,k_3} & & & A_3^{k_2,k_3} & \\ \vdots & & & & \ddots & \\ B_3^{k_2,K_3} & & & & & A_3^{k_2,K_3} \end{pmatrix} \tag{14}$$

$$f_2^{k_2} := \begin{pmatrix} b_2^{k_2} \\ b_3^{k_2,1} \\ \vdots \\ b_3^{k_2,k_3} \\ \vdots \\ b_3^{k_2,K_3} \end{pmatrix} \qquad \rho_2^{k_2} := \begin{pmatrix} c_2^{k_2} \\ p_3^1 c_3^{k_2,1} \\ \vdots \\ p_3^{k_3} c_3^{k_2,k_3} \\ \vdots \\ p_3^{K_3} c_3^{k_2,K_3} \end{pmatrix} \qquad z_2^{k_2} := \begin{pmatrix} x_2^{k_2} \\ x_3^{k_2,1} \\ \vdots \\ x_3^{k_2,k_3} \\ \vdots \\ x_3^{k_2,K_3} \end{pmatrix} \tag{15}$$

this leads to

$$\begin{aligned} &\min_{x_1}\{c_1 x_1 + \mathbb{E}_{\xi_2}\{\min_{\mathbf{z}_2} \rho_{\mathbf{2}}\mathbf{z}_2\} \\ s.t.\ & A_1 x_1 && = b_1 \\ & \mathbf{F}_2\mathbf{z}_1 \quad + \quad \mathbf{E}_2\mathbf{z}_2 && = \mathbf{f_2}\ a.s.\ , \end{aligned} \tag{16}$$

another two–stage problem with the last two stages combined. So, a three–stage problem can be converted to a two–stage problem by combining the last two–stages.

Thus aggregation can be applied to any multistage stochastic LP problem to obtain the same problem with a different number of time periods and a different size of subproblem. The original aggregation implementation in MSLiP by Gassmann allowed combining from the second stage to the end with a constant number of time stages being aggregated. This has been generalised to allow combining of any set of adjacent time stages together.

## Serial computation

MSLiP and MSLiP–OSL have been tested on a number of problem sets — all expressed in SMPS format [7]— see Tables 1, 2 and 3. These tables give the number of time periods, and the number of scenarios followed by the number of columns, rows, and non–zero elements in the deterministic equivalent problem. The last three columns of the tables give the optimal objective value, the branching structure of the tree (starting from the root node), and the EVPI value in terms of percentage of the optimal objective value, termed *stochasticity.* The EVPI values for a large number of these problem sets are extremely small, suggesting that these problems are not truly stochastic (SCSD8, SCAGR7, SCTAP1, SCFXM1, PLTEXP and STORM). There are other problem sets with more reasonable EVPI values (LOUV, FOREST, SGPF). The WATSON problem set has fairly high EVPI, while the SC205 set has extremely high EVPI, an artifact of the optimal value being 'close' to zero.

The branchings of the trees show unusual structures in some cases as well (the multi-stage versions of SC205, SCSD8, SCTAP1, SCFXM1). It is to be expected that either the branching at each stage of the tree should be fairly constant, or initial branching should be greater than later branching, as in practical problems the information which is closer to the initial time period is more relevant than that further into the future. Table 4 gives the nodal dimensions of subproblems.

Solutions have been obtained for all the problems by MSLiP–OSL and for the majority by MSLiP. Further experimentation with WATSON, the portfolio management test set can be found in [13], where nested Benders decomposition is compared favourably with both the simplex and interior point methods. Results for an IBM RS6000/590 with 128 MB of RAM running under AIX.3.2.5 are shown in Tables 5 and 6. The default setting for both MSLiP and MSLiP–OSL is single cuts for a two–stage problem and multicuts for multistage problems. The tables give the number of cuts and iterations taken to reach the solution when solving with MSLiP. The number of cuts and iterations taken by MSLiP–OSL is similar, although usually slightly greater due to the increased accuracy of the solver. The bunching routine was switched off in these experiments, to give a fair comparison of the solvers. With bunching switched on, problems such as PLTEXP can be solved much faster. This again suggests that many nodes have the same optimal basis, and therefore the problem has very low stochasticity. It should be noted that the stochasticity must be only in the right hand side of the problem if bunching is to be applicable. Truly stochastic problems such as financial portfolio management problems usually do not have this structure.

MSLiP has difficulties with multistage problems where many cuts are produced, and with problems with large nodes in their tree, such as STORM. The quickest solution time for the STORM set was obtained with MSLiP–OSL, with the presolve (PRESOLVE 3) switched on. Generally, however, presolving slows the solution time because the OSL presolver writes to disk, and the nodal problems are not large enough to warrant presolving. All the other MSLiP–OSL results presented are with the presolver off.

We have experimented with the nodal problem size for the multistage sets, in order to test the effects of having larger nodes and less stages on:

1. the solution time,
2. the effectiveness of obtaining a solution with nested Benders.

Aggregation results for the WATSON and FOREST problem sets, solving with MSLiP–OSL, can be found in Table 7, where it can be seen that larger nodal subproblems and fewer stages can improve the solution time for problems where the original nodes are extremely small. The time to solve WATSON.10.256 can be reduced from 143.8 seconds to 77.9 seconds for example. More information is gained from each node, so less cuts and iterations are needed to find the solution. If however, too many stages are combined, the solution time will be slowed. It appears that there is a preferable nodal subproblem size. An estimate of the nodal sizes would be between 100 — 300 rows and columns and this should decrease in size as the tree is descended. Problems, such as WATSON, in which later stages have larger subproblems than earlier stages appear to be more difficult to solve. (The Pfefferkorn–Tomlin

LP solver is extremely slow at solving these larger nodal subproblems and aggregation may not be effective with the MSLiP code. Modern LP solvers will not have the same difficulties as the Pfefferkorn–Tomlin LP solver.)

Changing the initial wealth of the WATSON problem set (see [13]) makes the feasiblity region small, and therefore creates a difficult problem for nested Benders decomposition to solve. Aggregation was tested on the effectiveness of obtaining solutions for such problems. Results for two problems, one with 1152 scenarios and the other with 256 scenarios, show how the structure of the problem may change the ease with which a problem is solved (Table 8).

These serial results show that nested Benders decomposition is a fast and effective method for solving multistage linear stochastic programming problems. An effective code needs a modern LP solver, such as OSL [30], CPLEX, [14], FORTMP [23] or XPRESSLP [15]. Changing the data structure of the problem through stage aggregation favouring earlier stages has been shown to lead to significant speedups in this algorithm.

## 4 Parallel algorithm and experiments

The *parallel algorithm* described below is more similar to the methods of [2] and [24] than to that of [8]. Other parallelization approaches to stochastic programming problems may be found in [16, 40]. It uses the *farming* technique, with a *master* and a certain number of *slaves*, decided by the user. The method can be described as follows

### Host

**Step H: 0** Send master module to processor 0.
Send slave module to processors $s,\ s = 1, 2, \ldots, S$.
Send data to processor 0.
**Step H: 4** Receive optimal solution from processor 0.

### Master processor

**Step M: 0** Receive module from host.
Receive data from host.
**Step M: 1** Solve master problem, obtaining trial solution.
**Step M: 2** Assuming there are $m \geq S$ nodes to be solved in the current time stage, send information for the first $S$ subproblems to slave processors $s,\ s = 1, 2, \ldots, S$.
**Step M: 3** When an optimal or infeasible solution is returned from a processor $k,\ k \in 1, 2, \ldots, S$, send the information to solve another node to processor $k$ unless there are no nodes left to be sent.
**Step M: 4** When all nodes have been solved, the master determines whether to move forward or backward in the tree. The master forms the cuts for all nodes.

On reaching the first stage in the tree and finding there are no more subproblems to be solved in the tree, the optimal solution has been reached, which is returned to the host, and the slave processors are informed to stop.
Otherwise return to **Step M: 2**.

## Slave processor

**Step S: 0** Receive module from host.
**Step S: 2** Receive and solve LP problem sent from master.
**Step S: 3** Return solution to master and return to **Step S: 2**.
**Step S: 4** Receive message from master to stop.

If the nodal subproblems are large enough this algorithm should have reasonable load–balancing properties, though there may be a communications bottleneck while slave processors try to communicate with the master.

The information required by a slave processor from the master is :–

1. New constraints (cuts).
2. New right hand side. (The master forms the right hand side).
3. Basis information.

The information required by the master processor from a slave is :–

1. New primal vector.
2. New dual vector.
3. Basis information.

The new constraints formed by the master are broadcast to all slaves, while all other information is passed directly between a slave and the master. Although this algorithm naturally load–balances, which is an advantage it has over the Birge *et al* method [9] (where unnecessary computation may be done on a processor) the amount of communication is relatively high, compared to calculation, when the nodes of the tree are relatively small, i.e the amount of communication in the Birge *et al* method is much less. The aggregation routine has been added to the parallel version of the MSLiP code in order to reduce the communication to calculation ratio, and reduce the number of cuts formed by the master. By Amdahl's law [1] a reduction in serial computation should increase the maximum possible speed–up.

As with the serial code, when a nodal problem is found to be infeasible and single cuts are being placed, no more sibling problems are solved. This is the case in the parallel code for multicuts as well. However the parallel code may lead to a different solution path. When a nodal problem is found to be infeasible, some of its siblings may be solved in the parallel algorithm which were not solved in the serial algorithm and this may lead to different cut information.

The above algorithm has been implemented by parallelizing the MSLiP code.

## Parallel results

The algorithm has been tested on a Meiko T800 Transputer array [48], on a network of IBM RS6000 25Ts using PVM and on a Fujitsu AP1000. The AP1000 processors are 5–7 MFLOP chips with 16MB of RAM. The communication bandwidth is between 5.6–25 Mb/s.

Parallel results for a subset of the problem sets are shown in Table 9 for the AP1000. Speed–up is defined to be the ratio of serial solution time to parallel solution time. Denoting the time to solution on $p$ processors by $T(p)$ and the *speed–up* on $p$ processors by $S(p)$, then $S(p) := T(1)/T(p)$. *Efficiency* $E(p)$ is the speed–up divided by the number of processors: $E(p) := S(p)/p$ . It can be seen that parallelization is effective for some problems while for others it has essentially no effect. This is dependent on the structure of the tree and the size of its nodes. The parallelization technique is more effective on problems which have the higher branching earlier in the tree and larger nodes.

Larger problems have not been solved due to the limited memory of the AP1000 processors. This unfortunately meant that the largest WATSON problem that could be solved had only 16 scenarios. Aggregation was attempted on the FOREST problem with 144 scenarios, but was again limited by memory, and therefore only the last few stages of the problem were aggregated. This, as previously noted, is not an efficient tree structure (in that the latter stages have larger nodal subproblems than earlier stages) and the serial solution time was slowed. The aggregation did however, have a significant effect on speed–up and efficiency (Table 10).

The three graphs in Figures 1 to 3 show the speed–up against the number of slave processors. Initially speed–up may be linear, but as the number of slave processors increases the increase in speed–up falls off, i.e. the efficiency of the method is poorer for more processors. On fewer processors the efficiency of the method is improved. This is due to:–

- A communications bottleneck developing, when slave processors try to return information to the master and receive another subproblem from it.
- The amount of time spent solving the root node and forming cuts on the master processor reducing the amount of speed–up possible (Amdahl's law).

The graphs can give an idea of the efficiency of the method on a certain number of processors. Where the speed–up is linear the efficiency is 1 (or alternatively 100 %).

Results for the transputer array [48] generally showed better levels of speed–up for the problems solved (the processors only had 4MB of RAM). This though is due to the fact that the calculation is very slow (3 MIPS) and therfore communication is less frequent. Problem sets such as LOUVEAUX showed impressive speed–up, whereas no speed–up was achieved for this problem set on the AP1000. Results on the network of workstations were poor, as the communication speed was slow in comparison to calculation. This greatly increased the solution time when parallelizing on one slave processor. Although the solution times then decreased as more slave processors were added, significant speed–ups were seldom achieved.

## 5 Conclusions and further work

This paper discusses nested Benders decomposition and its implementation in the MSLiP code as well as in the hybrid code MSLiP–OSL. It has been shown to be an effective solution method for multistage stochastic linear programming problems, which have many real applications, especially in finance. This decomposition method has been shown to outperform other solution methods for portfolio mangement problems [13].

In this paper, further improvements to the method have been made through aggregation and parallelization. The structure of the tree and the size of the nodes of the tree have been shown to be critical to improvements in solution time. An optimal structure may be constructed through stage aggregation for a particular model. Parallelization of the MSLiP code can improve the solution time too. This effect is again dependent on the nodal problem size as well as the feasibility tightness in terms of the number of iterations and cuts required. Aggregation was shown to be capable of improving parallelization efficiency further.

However aggregation has not been tested fully on the problem sets available and needs to be tried on more multistage problems. In order to achieve this, a parallel system with more RAM per processor is needed. We are currently experimenting with an IBM SP2 which has 16 nodes, each containing an RS6000 390 chip with 128 MB of RAM, and a total of 4GB of hard disk. The bandwith for this machine has a theoretical peak of 40Mb/s which is expected to reduce communication problems. The processors, at 128 MFLOPS (peak for double precision), are considerably faster than the AP1000 as well.

Speed–up can be improved by performing the formation of right hand sides and cuts on the slave processor, as this reduces the time spent in the serial mode of the algorithm. However, in the case of cut formation this would increase communication due to the need to broadcast cuts to ancestor subproblems running on undesignated processors. Other ideas for improving the algorithm are:–

- Solving more than one node at a time on a slave processor
- Solving a node and its children on a slave processor.

The second idea can be seen to be a cross–over between the method employed in this algorithm and that used by Birge *et al* [8]. Some of the above suggestions may be extremely difficult to implement in the MSLiP code and we are currently designing a new nested Benders code to overcome these difficulties.

Parallelization of the EVPI routine of MSLiP is currently in progress and should improve its calculation time dramatically in the multistage case, as this routine parallelizes more efficiently than nested Benders decomposition. Once this is achieved a parallel version of the Corvera–Dempster EVPI–based sampling algorithm [11] may be implemented which promises to make previously intractable financial portfolio management problems tractable (see [13]).

## 6 Acknowledgements

We are grateful to Dr. A. C. Downton, Department of Electronic Systems Engineering, University of Essex; Fujitsu (UK) Limited and the Imperial College Fujitsu Parallel Computing Research Centre for access to and support for the parallel computing facilities used in the experiments reported here.

## References

[1] G.M. Amdahl. Validity of the single processor approach to achieving large scale computing abilities. *AFIPS Conference Proceeding* **30** (1967) 483–485.

[2] K.A. Ariyawansa & D.D. Hudson. Performance of a benchmark parallel implementation of the Van–Slyke and Wets algorithm. *Concurrency: Practice and Experience* **3** (1991) 109–128.

[3] A.J. Berger, J.M. Mulvey , E. Rothberg & R. Vanderbei. Solving multistage stochastic programs using tree dissection. Research Report, Department of Civil Engineering and Operations Research, Princeton University (June 1995).

[4] J.F. Benders. Partitioning procedures for solving mixed–variable programming problems. *Numerische Mathematik* **4** (1962) 238–252.

[5] D.P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods.* Academic Press, New York (1982).

[6] D. Bienstock & J.F. Shapiro. Optimizing resource acquisition decisions by stochastic programming. *Management Science* **34** (1988) 215–229.

[7] J.R. Birge, M.A.H. Dempster, H.I. Gassmann, E.A. Gunn, A.J. King & S.W. Wallace. A standard input format for multiperiod stochastic linear programs. Mathematical Programming Society *Committee on Algorithms Newsletter* **17** (1988) 1–19.

[8] J.R. Birge, C.J. Donohue, D.F .Holmes, O.G. Svintsitski. A parallel implementation of the nested decomposition algorithm for multistage stochastic linear programs. Technical Report 94–1. Department of Industrial and Operations Engineering, University of Michigan (January 1994).

[9] J.R. Birge, C.J. Donohue, D.F .Holmes & O.G. Svintsitski. ND–UM Version 1.0 Computer Code for the Nested Decomposition Algorithm. Department of Industrial and Operations Engineering, University of Michigan (August 1994).

[10] S.P. Bradley & D.B. Crane. A dynamic model for bond portfolio management. *Management Science* **19** (1972) 139–151.

[11] X. Corvera Poiré. Model Generation and Sampling Algorithms for Dynamic Stochastic Programming. PhD Thesis, Department of Mathematics, University of Essex (August 1995).

[12] D.R. Carino & W.T. Ziemba. Formulation of the Russell–Yasuda Kasai financial planning model. Frank Russell Company, Tacoma, Washington (May 1995).

[13] G. Consigli & M.A.H. Dempster. Stochastic programming techniques for dynamic portfolio management. Submitted to *Annals of OR* (1995).

[14] CPLEX Optimization, Inc. Using the CPLEX Callable Library Version 3.0. Incline Village, Nevada (1994).

[15] Dash Associates Ltd. A quick tour of XPRESS–MP for Windows Version 1.0. Northants, England (August 1995).

[16] G.B. Dantzig, J.K. Ho & G. Infanger. Solving stochastic linear programs on a hypercube multicomputer. Technical Report SOL 91–10, Department of Operations Research, Stanford University ( August 1991).

[17] G.B. Dantzig & P. Wolfe. Decomposition principle for linear programs. *Operations Research* **8** (1960) 101–111.

[18] M.A.H.Dempster (ed.). *Stochastic Programming*. Academic Press, London (1980).

[19] M.A.H.Dempster. On stochastic programming II: Dynamic problems under risk. *Stochastics* **25** (1988) 15–42.

[20] M.A.H. Dempster & A.M. Ireland. A financial expert decision support system In: G. Mitra (ed.). *Mathematical Models for Decision Support* , NATO ASI Series F48. Springer–Verlag, Heidelberg (1988) 631-640.

[21] M.A.H. Dempster & A.M. Ireland. Object–oriented model integration in a financial decision support system. *Decision Support Systems* **7** (1991) 1–12.

[22] J. Dupačová. Multistage stochastic programs: The state–of–the–art and selected bibliography. *Kybernetika* **31** (1995) 151–174.

[23] E.F.D. Ellison, M. Hajian, R. Levkovitz, I. Maros, G. Mitra & D. Sayers. FORTMP Manual. Department of Mathematics and Statistics, Brunel University (1994).

[24] R. Entriken. The parallel decomposition of linear programs. Technical Report SOL 89–17, Department of Operations Research, Stanford University (November 1989).

[25] Yu. Ermoliev & R.J-B. Wets (eds.). *Numerical Techniques for Stochastic Optimization*, Springer–Verlag, Berlin (1988).

[26] Fujitsu Ltd. Special issue on cellular array processor AP1000. *Scientific and Technical Journal* **29** (1993).

[27] H.I Gassmann. MSLiP: A computer code for the multistage stochastic linear programming problem. *Mathematical Programming* **47** (1990) 407–423.

[28] H.I Gassmann. MSLiP 8.2 User's Guide. Working Paper, School of Business Administration, Dalhousie University (September 1992).

[29] P. Hutchinson & M. Lane. A model for managing a certificate of deposit portfolio under uncertainty. In: [18] 473–496.

[30] IBM Corporation. Optimization Subroutine Library (OSL) Release 2. Fourth Edition. Kingston, New York (1992).

[31] P. Kall & S.W. Wallace. *Stochastic Programming.* John Wiley and Sons, Chichester (1994).

[32] Kallberg & W. Ziemba. An algorithm for portfolio revision: Theory, computational algorithm and empirical results. In: R.L. Shultz (ed.). *Applications of Management Science*, Vol.I, JAI Press, Greenwich, CT (1981) 267–291.

[33] J.E. Kelley. The cutting–plane method for solving convex programs. *Journal of Social and Industrial Applied Mathematics* **7** (1960) 703–712.

[34] A. King. SP/OSL Version 1.0. Stochastic Programming Interface User's Guide. IBM Research Division, Yorktown Heights, New York (1994).

[35] M.I. Kusy & W.T. Ziemba. A bank asset and liability model. *Operations Research* **34** (1986) 356–376.

[36] R. Levkovitz & G. Mitra. Solution of large–scale linear programs: A review of hardware, software and algorithmic issues. In: T.A. Ciriani & R.C. Leachman (eds.). *Optimization in Industry.* John Wiley and Sons (1993) 139–171.

[37] D.G. Luenberger. *Linear and Nonlinear Programming.* Addison–Wesley, Reading, MA (1994).

[38] I.J. Lustig. R.E. Marsten & D.F. Shanno. Interior point methods for linear programming: Computational state of the art. *ORSA Journal on Computing* **6** (1994) 1-14.

[39] J.M. Mulvey. Integrative asset–liability planning using large–scale stochastic optimization. Report SOR 8, Department of Civil Engineering and Operations Research, Princeton University (July 1992).

[40] S.S Nielson & S.A. Zenios. A massively parallel algorithm for nonlinear stochastic network problems. Report 90–09–08, Decision Sciences Department, The Wharton School, University of Pennsylvania (October 1990)

[41] M.V.F. Pereirea & L.M.V.G. Pinto. Multi–stage stochastic optimization applied to energy planning. *Mathematical Programming* **52** (1991 )359–375 .

[42] C.E. Pfefferkorn & T.A. Tomlin. Design of a linear programming system for ILLIAC —V. Technical Report SOL 76–8, Department of Operations Research, Stanford University and Technical Report 5487, NASA–Ames Institute for Advanced Computation, Sunnyvale, CA (1976).

[43] A. Ruszczyński. Parallel decomposition of multistage stochastic programming problems. Research Report, Institute of Automatic Control, Warsaw University of Technology, 00665, Warsaw, Poland (August 1991).

[44] A. Ruszczyński. On augmented Lagrangian methods for multistage stochastic programs. Working Paper 94–05, International Institute for Applied Systems Analysis, Austria (February 1994)

[45] L. Somlyódy & R.J-B. Wets. Stochastic optimization models for lake eutrophication management. *Operations Research* **36.5** (1988) 660–681.

[46] U.H. Suhl. MOPS – Mathematical Optimization System. *European Journal of Operational Research* **72** (1994) 312–322.

[47] R. T. Thompson. MSLiP–OSL 8.3. User's Guide. Department of Mathematics, University of Essex (1995), forthcoming.

[48] R. T. Thompson. Implementation of nested Benders decomposition on an array of transputers. Working Paper 93–8. Department of Mathematics, University of Essex (November 1993).

[49] R.J. Vanderbei. LOQO User's Manual. Report SOR–12, Department of Civil Engineering and Operations Research, Princeton University (1993).

[50] R. Van Slyke & R. J-B Wets. L–shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal of Applied Mathematics* **17** (1969) 638–663.

[51] R.J-B. Wets. Large scale linear programming techniques. In: [25] 65–94.

[52] R. Wittrock. Dual nested decomposition of staircase linear programs. *Mathematical Programming Study* **24** (1985) 65–86.

[53] D. Yang & S.A. Zenios. Stochastic linear programming and robust optimization. Report 95–07. Department of Public and Business Administration, University of Cyprus, Cyprus (February 1995).

| Name | Per | # Scen | Col | Row | Nonzeros | Obj.val. | Tree | %EVPI |
|---|---|---|---|---|---|---|---|---|
| SC205 .2.200 | 2 | 200 | 4414 | 4412 | 12030 | -10.07 | 200 | 359.8 |
| SC205.2.800 | 2 | 800 | 17614 | 17612 | 48030 | -10.07 | 800 | 398.7 |
| SC205.2.1600 | 2 | 1600 | 35214 | 35212 | 96030 | 0.0 | 1600 | - |
| SC205.3.200 | 3 | 200 | 2258 | 2256 | 6149 | -10.07 | 4.50 | 359.8 |
| SC205.3.800 | 3 | 800 | 8902 | 8900 | 24269 | -10.07 | 8.100 | 398.7 |
| SC205.3.1600 | 3 | 1600 | 17790 | 17788 | 48509 | 0.00 | 16.100 | - |
| SCSD8.2.216 | 2 | 216 | 30310 | 4330 | 95170 | 23.40 | 216 | 0.33 |
| SCSD8.2.432 | 2 | 432 | 60550 | 8650 | 190210 | 25.80 | 432 | 0.24 |
| SCSD8.3.216 | 3 | 216 | 15610 | 2230 | 48970 | 23.40 | 6.36 | 0.33 |
| SCSD8.3.432 | 3 | 432 | 30730 | 4390 | 96490 | 25.80 | 6.72 | 0.24 |
| SCAGR7.3.432 | 2 | 432 | 17300 | 16431 | 54474 | -834193.83 | 432 | 0.01 |
| SCAGR7.2.864 | 2 | 864 | 34580 | 32847 | 108906 | -834446.93 | 864 | 0.01 |
| SCAGR7.3.432 | 3 | 432 | 8780 | 8337 | 27636 | -834107.52 | 6.72 | 0.02 |
| SCAGR7.3.864 | 3 | 864 | 17420 | 16545 | 54852 | -834446.93 | 6.144 | 0.02 |
| SCTAP1.2.216 | 2 | 216 | 20784 | 12990 | 76140 | 249.00 | 216 | 0.20 |
| SCTAP1.2.480 | 2 | 480 | 46128 | 28830 | 169068 | 248.50 | 480 | 0.20 |
| SCTAP1.3.216 | 3 | 216 | 10704 | 6690 | 39180 | 249.00 | 3.36 | 0.20 |
| SCTAP1.3.480 | 3 | 480 | 23376 | 14610 | 85644 | 248.50 | 6.80 | 0.20 |

Table 1: Problem characteristics - SC205, SCSD8, SCAGR7 and SCTAP1

| Name | Per | # Scen | Row | Col | Nonzeros | Obj.val. | Tree | %EVPI |
|---|---|---|---|---|---|---|---|---|
| SCFXM1.2.64 | 2 | 64 | 14514 | 9564 | 53799 | 2891.71 | 64 | 0.00 |
| SCFXM1.2.256 | 2 | 256 | 57714 | 37980 | 213159 | 2891.71 | 256 | 0.00 |
| SCFXM1.4.64 | 4 | 64 | 5874 | 4104 | 19299 | 2891.71 | 4.1.16 | 0.00 |
| SCFXM1.4.256 | 4 | 256 | 22002 | 15412 | 70559 | 2891.71 | 8.1.32 | 0.00 |
| LOUV.2.320 | 2 | 320 | 5124 | 2242 | 10248 | 480.04 | 320 | 8.87 |
| LOUV.2.640 | 2 | 640 | 10244 | 4482 | 20488 | 482.19 | 640 | 9.26 |
| LOUV.2.1280 | 2 | 1280 | 20484 | 8962 | 40968 | 482.64 | 1280 | 9.35 |
| FOREST.8.144 | 8 | 144 | 5679 | 5961 | 23839 | -43380.77 | $3^2.2^4.1$ | 3.48 |
| FOREST.8.288 | 8 | 288 | 11215 | 11771 | 47163 | -43663.28 | $4.3^2.2^3.1$ | 2.95 |
| FOREST.8.384 | 8 | 384 | 14927 | 15667 | 62795 | -43758.52 | $4^2.3.2^3.1$ | 2.75 |
| FOREST.8.512 | 8 | 512 | 19791 | 20771 | 83307 | -43868.95 | $4^3.2^3.1$ | 2.51 |
| PLTEXPA2.16 | 2 | 16 | 4540 | 1726 | 9233 | -9.6633 | 16 | 0.01 |
| PLTEXPA3.36 | 3 | 36 | 11612 | 4430 | 23611 | -13.9694 | $6^2$ | 0.00 |
| PLTEXPA3.256 | 3 | 256 | 74172 | 28350 | 150801 | -14.2675 | $16^2$ | 0.00 |
| PLTEXPA4.216 | 4 | 216 | 70364 | 26894 | 143059 | -19.5994 | $6^3$ | 0.00 |
| PLTEXPA4.4096 | 4 | 4096 | 1214492 | 454334 | – | -18.8493 | $16^3$ | 0.00 |
| PLTEXPA5.1296 | 5 | 1296 | 432200 | 161678 | – | -23.2141 | $6^4$ | 0.00 |

Table 2: Problem characteristics - SCFXM1, LOUV, FOREST and PLTEXP

| Problem | Per | # Scen. | Col | Row | Nonzeros | Obj. Value | Tree | %EVPI |
|---|---|---|---|---|---|---|---|---|
| STORMG2.8 | 2 | 8 | 10193 | 4409 | 27424 | 15535235.730 | 8 | 0.03 |
| STORMG2.27 | 2 | 27 | 34114 | 14441 | 90903 | 15508982.306 | 27 | 0.21 |
| STORMG2.125 | 2 | 125 | 157496 | 66185 | 418321 | 15512090.180 | 125 | 0.23 |
| STORMG2.1000 | 2 | 1000 | 1387360 | 350185 | – | 15802589.698 | 1000 | 0.23 |
| SGPF3Y3 | 3 | 25 | 1617 | 1208 | 4090 | -2967.917 | $5^2$. | 4.08 |
| SGPF3Y5 | 5 | 625 | 39867 | 30458 | 103090 | -5172.165 | $5^4$ | 4.28 |
| SGPF3Y6 | 6 | 3125 | 199341 | 152434 |  | -6463.323 | $5^5$ | 5.97 |
| SGPF5Y3 | 3 | 25 | 2509 | 1952 | 6570 | -3027.706 | $5^2$ | 11.07 |
| SGPF5Y5 | 5 | 625 | 61759 | 49202 | 165570 | -5201.282 | $5^4$ | 10.84 |
| WAT.10.16 | 10 | 16 | 8401 | 4573 | 21368 | -2158.75 | $2^4.1^5$ | 12.64 |
| WAT.10.64 | 10 | 64 | 28097 | 15101 | 72648 | -2310.53 | $2^6.1^3$ | 30.43 |
| WAT.10.128 | 10 | 128 | 49153 | 26237 | 128648 | -1637.81 | $2^7.1^2$ | 42.75 |
| WAT.10.256 | 10 | 256 | 82177 | 43517 | 218888 | -2000.84 | $2^8.1$ | 36.63 |
| WAT.10.512 | 10 | 512 | 128001 | 67069 | 350728 | -1959.63 | $2^9$ | 44.63 |
| WAT.10.1024 | 10 | 1024 | 255987 | 134127 | 701428 | -1926.79 | $4.2^8$ | 46.53 |

Table 3: Problem characteristics - STORM, SGPF and WATSON

| Name | Per 1 (RowsxCols) | Per 2 (RowsxCols) | Per 3 (RowsxCols) | Per 4 (RowsxCols) | Per 5 (RowsxCols) |
|---|---|---|---|---|---|
| SC205.2 | (12x14) | (22x22) | - | - | - |
| SCSD8.2 | (10x70) | (20x140) | - | - | - |
| SCAGR7.2 | (15x20) | (38x40) | - | - | - |
| SCTAP1.2 | (30x48) | (60x96) | - | - | - |
| SCFXM1.2 | (92x114) | (148x225) | - | - | - |
| LOUV.2 | (2x4) | (7x16) | - | - | - |
| PLTEXP.2 | (62x188) | (104x272) | - | - | - |
| STORM.2 | (185x121) | (528x1259) | - | - | - |
| SC205.3 | (12x14) | (11x11) | (11x11) | - | - |
| SCSD8.3. | (10x70) | (10x70) | (10x70) | - | - |
| SCAGR7.3 | (15x20) | (19x20) | (19x20) | - | - |
| SCTAP1.3 | (30x48) | (30x48) | (30x48) | - | - |
| PLTEXP.3 | (62x188) | (104x272) | (104x272) | - | - |
| SGPF.3y3 | (38x87) | (39x51) | (39x51) | - | - |
| SGPF.5y3 | (62x139) | (63x79) | (63x79) | - | - |
| SCFXM1.4 | (92x114) | (82x99) | (9x45) | (57x81) | - |
| PLTEXP.3 | (62x188) | (104x272) | (104x272) | (104x272) | - |
| SGPF.3y5 | (38x87) | (39x51) | (39x51) | (39x51) | (39x51) |
| SGPF.5y5 | (62x139) | (63x79) | (63x79) | (63x79) | (63x79) |

| Name | Per 1, 2 (RowsxCols) | Per 3, 4 (RowsxCols) | Per 5, 6 (RowsxCols) | Per , 7, 8 (RowsxCols) | Per 9, 10 (RowsxCols) |
|---|---|---|---|---|---|
| FOREST.8 | (15x15),(17x16) | (17x16),(17x16) | (17x16),(17x16) | (17x16),(8x8) | |
| WAT.10 | (11x15),(15x23) | (19x31),(23x39) | (27x47),(31x55) | (35x63),(39x71) | (43x79),(92x179) |

Table 4: Problem characteristics - Nodal dimensions

| Name | MSLiP CPU | -OSL CPU | Itns | Cuts (Feas) |
|---|---|---|---|---|
| SC205 .2.200 | 1.6 | 5.5 | 20 | 19(18) |
| SC205.2.800 | 10.5 | 35.5 | 27 | 26(25) |
| SC205.2.1600 | 32.9 | 97.9 | 32 | 31(30) |
| SC205.3.200 | 0.3 | 1.1 | 7 | 18(13) |
| SC205.3.800 | 1.3 | 4.4 | 7 | 30(21) |
| SC205.3.1600 | 2.4 | 11.5 | 8 | 55(38) |
| SCSD8.2.216 | 7.5 | 3.3 | 2 | 1(0) |
| SCSD8.2.432 | 14.0 | 6.6 | 2 | 1(0) |
| SCSD8.3.216 | 1.1 | 3.8 | 4 | 12(0) |
| SCSD8.3.432 | 3.2 | 7.4 | 5 | 20(0) |
| SCAGR7.3.432 | 21.7 | 88.7 | 13 | 12(3) |
| SCAGR7.2.864 | 30.5 | 89.3 | 8 | 7(3) |
| SCAGR7.3.432 | 9.2 | 39.9 | 16 | 105(14) |
| SCAGR7.3.864 | 9.0 | 39.7 | 7 | 42(14) |
| SCTAP1.2.216 | 12.3 | 7.2 | 3 | 2(0) |
| SCTAP1.2.480 | 27.4 | 15.9 | 3 | 2(0) |
| SCTAP1.3.216 | 3.1 | 7.8 | 4 | 17(0) |
| SCTAP1.3.480 | 6.9 | 23.7 | 4 | 17(0) |

| Name | MSLiP CPU | -OSL CPU | Itns | Cuts (Feas) |
|---|---|---|---|---|
| SCFXM1.2.64 | 75.9 | 76.6 | 77 | 76(66) |
| SCFXM1.2.256 | 295.7 | 290.1 | 77 | 76(66) |
| SCFXM1.4.64 | 16.4 | 19.5 | 180 | 251(233) |
| SCFXM1.4.256 | 34.9 | 61.7 | 180 | 323(307) |
| LOUV.2.320 | 11.5 | 40.6 | 41 | 40(0) |
| LOUV.2.640 | 28.11 | 103.4 | 56 | 55(0) |
| LOUV.2.1280 | 62.3 | 226.5 | 47 | 46(0) |
| FOREST.8.144 | 15.9 | 59.5 | 20 | 3966(18) |
| FOREST.8.288 | 23.7 | 135.4 | 19 | 6095(10) |
| FOREST.8.384 | 33.3 | 169.5 | 18 | 8637(16) |
| FOREST.8.512 | 41.1 | 225.3 | 18 | 10446(18) |
| PLTEXPA2.16 | 6.0 | 0.7 | 2 | 1(0) |
| PLTEXPA3.36 | 15.9 | 2.2 | 2 | 7(0) |
| PLTEXPA3.256 | 98.2 | 11.9 | 2 | 17(0) |
| PLTEXPA4.216 | 114.1 | 12.0 | 2 | 43(0) |
| PLTEXPA4.4096 | 1644.87 | 184.9 | 2 | 273(0) |
| PLTEXPA5.1296 | 610.5 | 69.1 | 2 | 259(0) |

Table 5: Serial solution times on an IBM RS6000/590 - no bunching

| Name | MSLiP CPU | -OSL CPU | Itns | Cuts (Feas) |
|---|---|---|---|---|
| STORMG2.8 | - | 131.3 | 40 | 39(0) |
| STORMG2.27 | - | 451.8 | 42 | 41(0) |
| STORMG2.125 | - | 3055.6 | 62 | 61(0) |
| STORMG2.1000 | - | 22071.9 | 56 | 55(0) |
| SGPF3Y3 | 0.7 | 0.3 | 2 | 6(0) |
| SGPF3Y5 | 18.2 | 18.4 | 3 | 198(0) |
| SGPF3Y6 | 105.2 | 191.3 | 5 | 1057(0) |
| SGPF5Y3 | 1.7 | 0.3 | 3 | 11(0) |
| SGPF5Y5 | 40.2 | 9.7 | 4 | 178(0) |

| Name | MSLiP CPU | -OSL CPU | Itns | Cuts (Feas) |
|---|---|---|---|---|
| WAT.10.16 | 7.6 | 7.6 | 6 | 306(0) |
| WAT.10.64 | 32.7 | 37.0 | 11 | 1170(1) |
| WAT.10.128 | 57.4 | 72.7 | 11 | 1492(0) |
| WAT.10.256 | 144.2 | 143.8 | 13 | 2291(13) |
| WAT.10.512 | 181.9 | 181.0 | 12 | 2885(0) |
| WAT.10.1024 | 371.3 | 369.8 | 15 | 7499(94) |

Table 6: Serial solution times on an IBM RS6000/590 - no bunching

| Name | Per 1 | Per 2 | Per 3 | Per 4 | Per 5 | Per 6 |
|---|---|---|---|---|---|---|
| Agg.1 | 1-2 | 3-4 | 5-6 | 7-8 | 9 | 10 |
| Agg.2 | 1-3 | 4-6 | 7-9 | 10 | | |
| Agg.3 | 1-3 | 4-6 | 7-8 | 9-10 | | |
| Agg.4 | 1-3 | 4-5 | 6-7 | 8-9 | 10 | |
| Agg.5 | 1-3 | 4-6 | 7 | 8 | 9 | 10 |
| Agg.6 | 1 | 2 | 3 | 4-5 | 6-8 | |

| Name | CPU | Itns | Cuts |
|---|---|---|---|
| WAT.16.Agg1 | 6.5 | 6 | 170(0) |
| WAT.16.Agg2 | 5.0 | 5 | 91(0) |
| WAT.16.**Agg3** | **4.9** | 5 | 95(0) |
| WAT.16.Agg4 | 5.1 | 5 | 136(0) |
| WAT.64.Agg1 | 31.4 | 8 | 569(0) |
| WAT.64.**Agg2** | **29.7** | 8 | 323 |
| WAT.64.Agg3 | 31.8 | 9 | 375(0) |
| WAT.64.Agg4 | 32.5 | 9 | 456(0) |
| FOREST.8.144.Agg1 | 56.7 | 12 | 1103(0) |
| FOREST.8.144.Agg2 | 51.0 | 9 | 694(0) |
| FOREST.8.144.**Agg4** | **43.5** | 11 | 691(0) |
| FOREST.8.288.Agg1 | 128.5 | 14 | 2135(0) |
| FOREST.8.288.Agg2 | 116.8 | 10 | 1450(0) |
| FOREST.8.288.**Agg4** | **91.8** | 11 | 11 |

| Name | CPU | Itns | Cuts |
|---|---|---|---|
| WAT.128.Agg1 | 75.0 | 11 | 859(0) |
| WAT.128.Agg2 | 58.0 | 8 | 445(0) |
| WAT.128.**Agg3** | **56.4** | 9 | 521(0) |
| WAT.128.Agg4 | 47.9 | 9 | 784(0) |
| WAT.256.Agg1 | 102.0 | 9 | 1464(2) |
| WAT.256.Agg2 | 119.4 | 8 | 712 |
| WAT.256.**Agg3** | **77.9** | 8 | 850(1) |
| WAT.256.Agg4 | 89.5 | 8 | 1101 |
| FOREST.8.384.Agg1 | 172.9 | 14 | 2860(0) |
| FOREST.8.384.Agg2 | 171.7 | 11 | 1928(0) |
| FOREST.8.384.**Agg4** | **128.4** | 11 | 1845(0) |
| FOREST.8.512.Agg1 | 214.6 | 13 | 3695(0) |
| FOREST.8.512.Agg2 | 232.4 | 11 | 2602(0) |
| FOREST.8.512.**Agg4** | **171.1** | 11 | 2575(0) |

Table 7: Aggregation on the FOREST and WATSON problem sets with MSLiP-OSL

| Name | CPU | Itns | Cuts |
|---|---|---|---|
| WAT.256 | Numerical errors | | |
| WAT.256.**Agg3** | **266.2** | 12 | 1117(11) |
| WAT.1152 | Numerical errors | | |
| WAT.1152.**Agg5** | **419.6** | 2 | 6023(113) |

| Name | CPU | Itns | Cuts |
|---|---|---|---|
| WAT.256 | 472.1 | 15 | 2085(60) |
| WAT.256.**Agg3** | **266.2** | 12 | 1117(11) |
| WAT.1152 | Numerical errors | | |
| WAT.1152.**Agg5** | **654.2** | 13 | 5556(52) |

Table 8: Aggregation to obtain a WATSON solution with - MSLiP and MSLiP-OSL

| Name | Serial CPU | # Processors | CPU | Speed-up | Efficiency % |
|---|---|---|---|---|---|
| SC205 .2.200 | 11.8 | 8 | 16.7 | 0.7 | 8.8 |
| SC205.3.200 | 2.1 | 8 | 5.8 | 0.4 | 5.0 |
| SCSD8.2.**216** | 57.2 | 24 | **6.6** | **8.5** | **35.4** |
| SCSD8.3.216 | 8.6 | 8 | 10.6 | 0.8 | 10.0 |
| SCAGR7.3.432 | 170.3 | 8 | 67.3 | 2.5 | 31.3 |
| SCAGR7.3.432 | 70.3 | 8 | 66.7 | 1.1 | 13.8 |
| SCTAP1.2.**216** | 96.0 | 15 | **14.4** | **6.6** | **44.0** |
| SCTAP1.3.216 | 24.8 | 8 | 11.2 | 2.2 | 27.5 |
| SCFXM1.2.**64** | 602.0 | 24 | **88.6** | **6.8** | **28.3** |
| SCFXM1.4.64 | 159.7 | 8 | 116.3 | 1.4 | 17.5 |
| LOUV.2.640 | 83.0 | 8 | 124.7 | 0.6 | 7.5 |
| FOREST.8.288 | 194.0 | 8 | 260.7 | 0.7 | 8.8 |
| PLTEXPA3.**36** | 200.3 | 15 | **48.1** | **4.1** | **27.3** |
| SGPF**5Y4** | 62.2 | 15 | **10.8** | **5.7** | **38.0** |
| WAT.10.16 | 64.4 | 15 | 21.5 | 2.9 | 14 |

Table 9: Parallel results on the AP1000

| Name | Serial CPU | # Processors | CPU | Speed-up | Efficiency |
|---|---|---|---|---|---|
| FOREST.8.144 | 139.4 | 15 | 99.2 | 1.4 | 9.3 |
| FOREST.8.144.Agg6 | 297.4 | 15 | 40.7 | 7.3 | 48.7 |

Table 10: Parallel Aggregation with FOREST.8.144

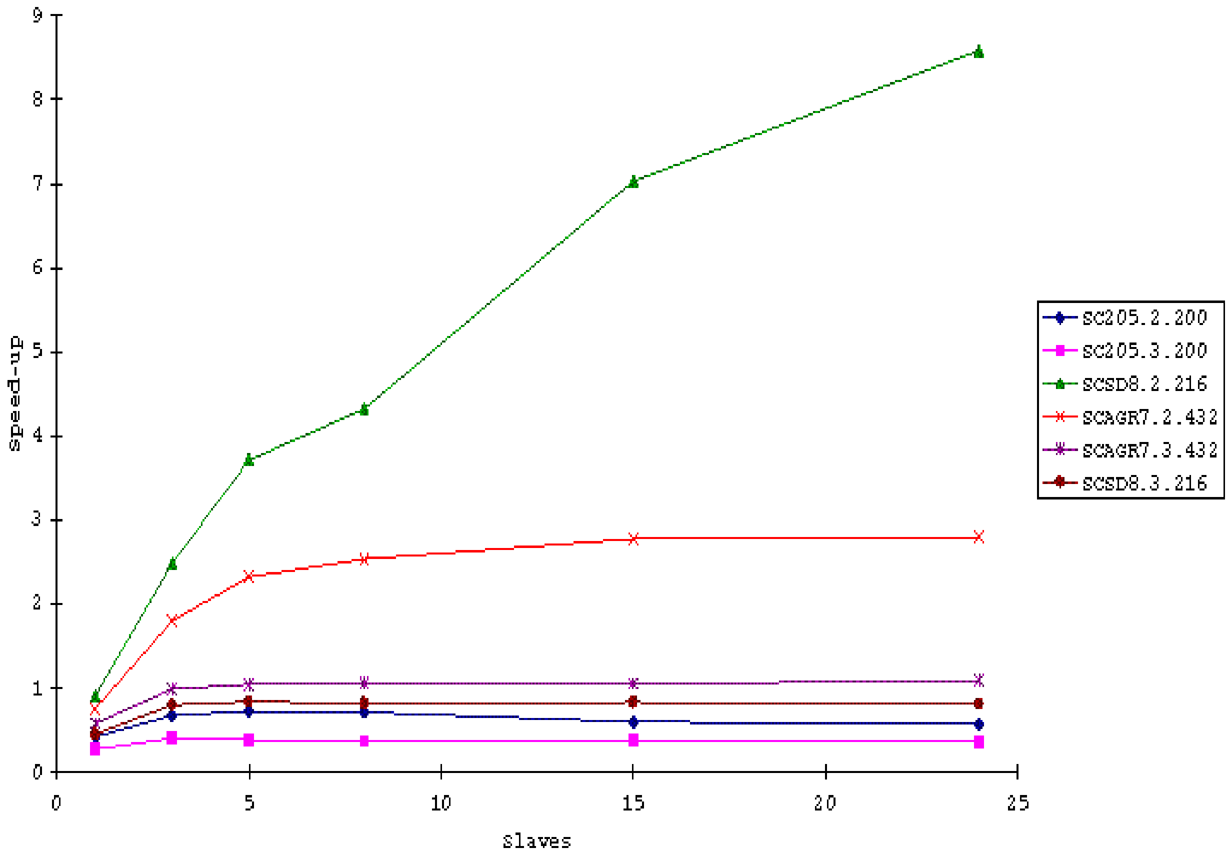

Figure 1: Results on the AP1000 for various problems

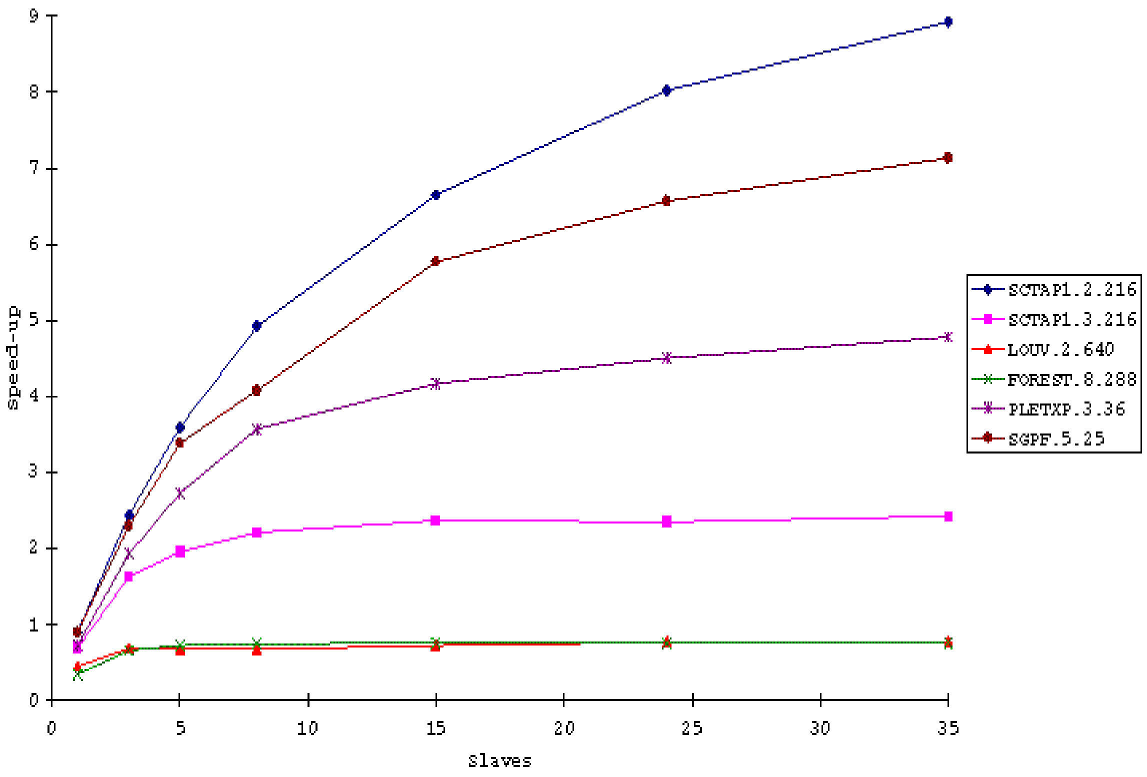

Figure 2: Results on the AP1000 for various problems

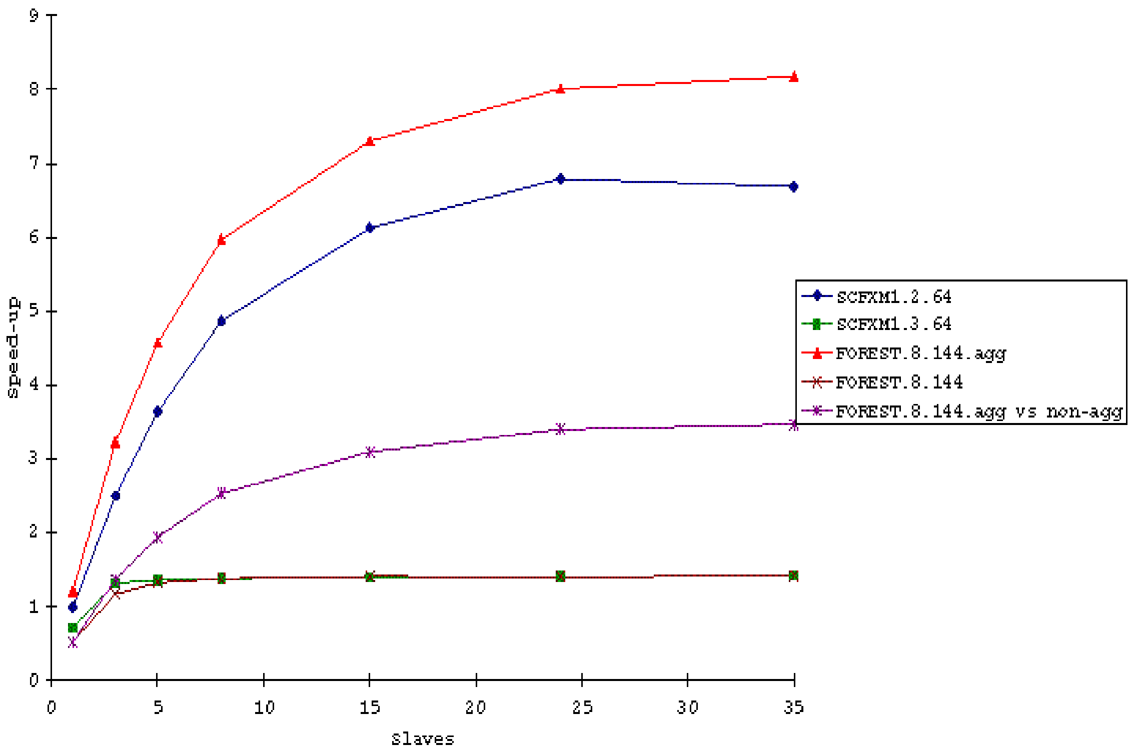

Figure 3: Results on the AP1000 for various problems