International Institute for Applied Systems Analysis
IIASA www.iiasa.ac.at

Parallel Decomposition of Multistage Stochastic Programming Problems

Ruszczynski, A.

IIASA Working Paper

WP-88-094

October 1988

WORKING PAPER

PARALLEL DECOMPOSITION OF MULTISTAGE STOCHASTIC PROGRAMMING PROBLEMS

A. Ruszczyński

October 1988
WP-88-094

IIASA
International Institute
for Applied Systems Analysis

PARALLEL DECOMPOSITION OF MULTISTAGE STOCHASTIC PROGRAMMING PROBLEMS

A. Ruszczyński

October 1988
WP-88-094

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
A-2361 Laxenburg, Austria

Foreword

A new decomposition method for multistage stochastic linear programming problems is proposed by the author. The method combines the ideas of the regularized decomposition method for two-stage programs and dynamic programming. With each node of the decision tree of the multistage stochastic problem a certain regularized subproblem is associated which generates decisions for its successors and some backward information for its predecessor. The subproblems are solved in parallel and exchange information in an asynchronous way through special buffers. After a finite time the method either finds an optimal solution to the problem or discovers its inconsistency. This method is especially convenient for implementation on a parallel computer.

Alexander B. Kurzhanski
Chairman
System and Decision Sciences Program

Parallel Decomposition of Multistage Stochastic Programming Problems

Andrzej Ruszczyński

Institute of Automatic Control

Warsaw University of Technology

00665 Warsaw, Poland

1 Introduction

The main objective of this paper is to present a parallel decomposition method for solving multistage stochastic linear programming problems defined as follows.

Let Ω be a finite probability space with elementary events ω and probabilities p_ω. Next, let $D_\omega(t)$ and $H_\omega(t)$, $t = 1, \ldots, T$ be sequences of random $m_b \times m_x$ matrices and $b_\omega(t)$ and $c_\omega(t)$, $t = 1, \ldots, T$, be sequences of random vectors in R^{m_b} and R^{m_x}, respectively. We shall call each sequence $b_\omega(t) = (D_\omega(t), H_\omega(t), b_\omega(t), c_\omega(t))$ corresponding to some event $\omega \in \Omega$ a *scenario*. The problem is to find a sequence $x_\omega(t)$, $t = 1, \ldots, T$, $\omega \in \Omega$, of random vectors in R^{m_x} (a *policy*), which minimizes the linear form

$$\sum_{\omega \in \Omega} p_\omega c_\omega(t)^* x_\omega(t) \tag{1.1}$$

subject to the constraints

$$D_\omega(t)x_\omega(t-1) + H_\omega(t)x_\omega(t) = b_\omega(t),\ t = 1, \ldots, T,\ \omega \in \Omega, \tag{1.2}$$

$$x_\omega(t) \geq 0,\ t = 1, \ldots, T,\ \omega \in \Omega, \tag{1.3}$$

$x(0) = x_0$, and an additional *nonanticipativity constraint*, which can be formulated as follows: for all $\omega^1, \omega^2 \in \Omega$ and any $t \in \{1, \ldots, T\}$

$$x_{\omega^1}(t) = x_{\omega^2}(t) \ \ \textit{if} \ \ s_{\omega^1}(\tau) = s_{\omega^2}(\tau) \ \ \textit{for} \ \ \tau = 1, \ldots, t. \tag{1.4}$$

In other words, decisions corresponding to scenarios which are indistinguishable up to time t should be equal (see [16] for an extensive discussion of this issue).

Two important special cases of (1.1)-(1.4) are the *deterministic control problem* (with one scenario) and the *two-stage stochastic programming problem* ($T = 2$, $s_\omega(1)$ identical for all $\omega \in \Omega$).

Although in principle (1.1)-(1.4) is a linear programming problem, its size may be too large for standard linear programming techniques [12]. For this reason a variety of specialized approaches have been developed for the two cases mentioned earlier.

The first group of methods are special versions of the simplex method which take advantage of the structure of the constraint matrix of the problem to improve basis factorization techniques and pricing strategies [3,6,7,10,13,21,19,23].

The second group are techniques coming down from the decomposition principle of Dantzig and Wolfe [2,4,5,9,22,23,24].

The third group are nonlinear methods specialized to this particular class of problems: the finite generation method [15], the progressive hedging algorithm [16] and the regularized decomposition method [17,18,20]. The latter one is of special interest for us, because it shares the finite convergence property of pure linear approaches.

The objective of our paper is twofold. First, we shall extend the regularized decomposition method to multistage stochastic programs, while retaining properties observed in the two-stage case. Secondly, we shall show that the subproblems into which (1.1)-(1.4) is decomposed can be solved in parallel and can exchange information in an asynchronous manner. We hope that this is of interest in its own right and brings new quality even to the earlier two-stage version of [17]: the subproblems *and* the master can operate in parallel. In the multistage case our approach may mitigate the effort required by nested formulations [2,9,24] by allowing fast transmission of information between the stages. For computers on which true multitasking is not yet possible our results eliminate restrictions on the order in which the subproblems are processed.

In section 2 we restate the problem in a tree-like form and give a general outline of the method. In section 3 we study in detail fundamental objects of our method: regularized subproblems and we describe how they generate information for the other subproblems. Section 4 contains a formal description of the method and in section 5 we prove its finite convergence.

2 Outline of the method

More insight into the structure of problem (1.1)-(1.4) can be gained by restating it in a tree-like form. Namely, with the set of scenarios $s_\omega(t)$, $t = 1,\dots,T$, $\omega \in \Omega$, we can associate a tree $\mathcal{T} = \{\mathcal{N}, \mathcal{A}\}$, where $\mathcal{N}$ is a set of nodes and $\mathcal{A}$ is a set of arcs of $\mathcal{T}$. The set of nodes $\mathcal{N}$ is divided into subsets (levels) $\mathcal{N}_t$, $t = 1,\dots,T$, and the nodes $n \in \mathcal{N}_t$ at level t correspond to different subscenarios $\{s^n(1),\dots,s^n(t)\}$. At level 1 there are so many nodes as many different realizations of $s(1)$ can occur; at level 2 the nodes correspond to different pairs $\{s(1), s(2)\}$, etc. The number of nodes at level T is equal to the number of scenarios $|\Omega|$. The arcs join nodes from neighboring levels in such a way that a node n at level t corresponding to subscenario $s^n = \{s^n(1),\dots,s^n(t)\}$ is connected with all nodes m at level $t+1$ whose subscenarios $s^m = \{s^m(1),\dots,s^m(t+1)\}$ equal s^n up to time t. Let us denote by $\pi(n)$ the *predecessor* of node n, i.e. the node at the previous level with which n is connected and by $\mathcal{S}(n)$ the set of *successors* of n, $\mathcal{S}(n) = \{m : n = \pi(m)\}$. Next, let $\Pi(n) = \{n, \pi(n), \pi(\pi(n)), ...\}$ be the *path* from n to level 1. Taking account of the fact that for $n \in \mathcal{N}_t$ we have $s^n = \{s^{\pi(n)}, s^n(t)\}$, it is sufficient to associate with each node $n \in \mathcal{N}_t$ only the last element of its subscenario, $s_n = s^n(t)$; the whole subscenario can be recovered by backtracking the path $\Pi(n)$.

A node n at level t corresponds to the bundle Ω_n of scenarios which are indistinguishable up to time t. By the nonanticipativity condition (1.4) all decisions $x_\omega(t)$, $\omega \in \Omega_n$ must be equal. We denote their value by x_n.

To complete the reformulation of the problem, with nodes $n \in \mathcal{N}$ we shall associate probabilities $\bar{p}_n$ defined as follows: for each *leaf* $n \in \mathcal{N}_T$ we set $\bar{p}_n = p_\omega$, where $\omega \in \Omega$ is the

event that corresponds to leaf n. For other nodes we define $\bar{p}_n = \sum_{m \in S(n)} \bar{p}_m$.

Using this notation we can rewrite (1.1)-(1.4) as follows:

$$\textit{minimize} \sum_{n \in \mathcal{N}} \bar{p}_n c_n^* x_n \tag{2.1}$$

$$D_n x_{\pi(n)} + H_n x_n = b_n,\ n \in \mathcal{N}, \tag{2.2}$$

$$x_n \geq 0,\ n \in \mathcal{N}, \tag{2.3}$$

where for $n \in \mathcal{N}_1$ we set $x_{\pi(n)} = x_0$. We shall assume throughout this paper that (2.1)-(2.3) is bounded.

The tree structure makes it possible to develop a hierarchical method for solving (2.1)-(2.3). For each pair of nodes (m, n), $m \in S(n)$, we define the conditional probability $p_{mn} = \bar{p}_m / \bar{p}_n$ and use it for recursive definition of the *value function*

$$f_n(x_{\pi(n)}) = \textit{min}\ \{c_n^* x_n + \sum_{m \in S(n)} p_{mn} f_m(x_n) :\ H_n x_n = b_n - D_n x_{\pi(n)},\ x_n \geq 0\}. \tag{2.4}$$

Solving the problem is equivalent to calculating

$$\sum_{n \in \mathcal{N}_1} \bar{p}_n f_n(x_0).$$

Since each component of this sum can be computed independently, with no loss of generality we can assume that there is only one node $n = 1$ at level 1 and our aim is to find $f_1(x_0)$. This can be done by the *nested decomposition method*: a recursive procedure of dynamic programming type in which problems (2.4) at various levels of recursion are solved by a cutting plane method (cf. [2,9,24]).

We shall modify the hierarchical approach in two directions.

First, instead of the pure cutting plane method we shall use its regularized version analysed in the two-stage case in [17]. With each node n of the tree $\mathcal{T}$, except for the leaves, we associate the following *regularized subproblem*

$$\textit{minimize}\ \ \eta_n = \tfrac{1}{2}\|x_n - \xi_n\|^2 + c_n^* x_n + \sum_{m \in S(n)} p_{mn} \bar{f}_m(x_n) \tag{2.5}$$

$$H_n x_n = b_n - D_n x_{\pi(n)}, \tag{2.6}$$

$$x_n \geq 0. \tag{2.7}$$

Here ξ_n is a certain regularizing point and $\bar{f}_m(\cdot)$, $m \in S(n)$ are convex piecewise linear outer approximations of the value functions $f_m(\cdot)$:

$$\bar{f}_m(x_n) \leq f_m(x_n)\ \ \textit{for all}\ \ x_n. \tag{2.8}$$

With each leaf $n \in \mathcal{N}_T$ of $\mathcal{T}$ we associate the linear problem

$$\textit{minimize}\ \ l_n = c_n^* x_n \tag{2.9}$$

$$H_n x_n = b_n - D_n x_{\pi(n)}, \tag{2.10}$$

$$x_n \geq 0. \tag{2.11}$$

In the method we link subproblems (2.5)-(2.7) and (2.9)-(2.11) in the same way in which the nodes of $\mathcal{T}$ are linked. They exchange information along the arcs by passing to their sons the

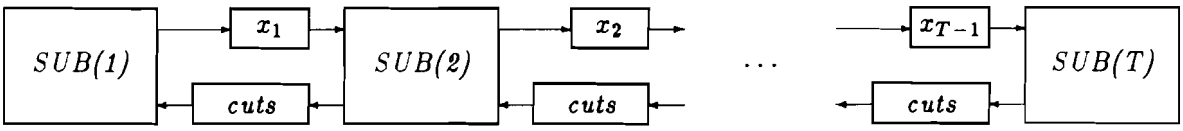

Figure 1: The network of tasks for the deterministic dynamic problem

solutions x_n and obtaining some backward information used to correct the approximations $\bar{f}_m(\cdot)$. The backward information has the form of *cuts*, i.e. some linear functions used to describe pieces of $\bar{f}_m(\cdot)$ or facets of their domains.

Our principal objective, however, is parallelization. In our method we allow *all* subproblems to be solved in a parallel asynchronous manner. Their logical dependence, implied by the tree structure of the problem, is reflected only in the communication structure of the distributed method, but does not condition the order in which the subproblems are processed. To this end we separate subproblems by buffers which store primal solutions passed from ancestor problems and cuts generated by the successors. Each subproblem takes some (possibly outdated) information from the buffers, generates its primal solution and and a cut, passes them to the neighboring buffers, etc., until no new information appears.

We shall discuss all these issues in sections 3 and 4, but let us at first illustrate the structure of the method on two typical examples.

Example 1. Consider the deterministic dynamic problem

$$\textit{minimize} \quad \sum_{t=1}^{T} c_t^* x_t$$

$$D_t x_{t-1} + H_t x_t = b_t, \; t = 1, \ldots, T,$$

$$x_t \geq 0, \; t = 1, \ldots, T.$$

Graph $\mathcal{T}$ is in this case a chain and the corresponding network of subproblems and buffers takes on the form shown in Figure 1. It corresponds to the nested decomposition method, but our subproblems are quadratic and solved in parallel thus allowing for fast exchange of information between the stages (see [1] for another parallel approach to dynamic programming).

Example 2. Consider now the stochastic two-stage problem

$$\textit{minimize} \quad c_1^* x_1 + \sum_{l=2}^{L} p_l c_l^* x_l$$

$$D_l x_1 + H_l x_l = b_l, \; l = 2, \ldots, L,$$

$$x_l \geq 0, \; l = 2, \ldots, L.$$

Graph $\mathcal{T}$ is a star with root 1 and leaves $2, \ldots, L$. The corresponding network of subproblems and buffers is shown in Figure 2. It is similar to the structure of the Dantzig-Wolfe method, but our master is different, and the master *and* the subproblems are solved in parallel, which significantly differs our approach from that of [8].

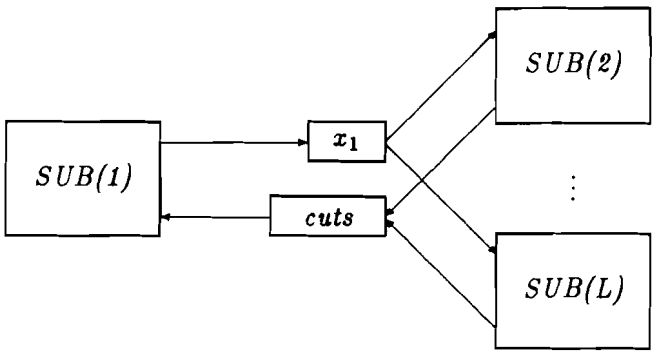

Figure 2: The network of tasks for the stochastic two-stage problem

For stochastic dynamic problems the structure of the network of subproblems is a combination of these two extreme cases.

3 Cuts

Let $a_{mj} + g^*_{mj}x_n$, $j \in J_m$, be a collection of linear functions such that

$$f_m(x_n) \geq a_{mj} + g^*_{mj}x_n, \quad \textit{for all} \quad x_n, \quad j \in J^1_m, \tag{3.1}$$

and

$$\text{dom } f_m \subseteq \{x_n : a_{mj} + g^*_{mj}x_n \leq 0\}, \quad j \in J^2_m, \tag{3.2}$$

where J^1_m and J^2_m are disjoint subsets of J_m. We shall call (3.1) *objective cuts* and (3.2) *feasibility cuts*. The cuts can be used to define functions $\bar{f}_m$ in (2.5) as follows: if x_n satisfies the feasibility cuts we set

$$\bar{f}_m(x_n) = \min\{v_{mn} : v_{mn} \geq a_{mj} + g^*_{mj}x_n, \; j \in J^1_m\};$$

otherwise we set $\bar{f}_m(x_n) = +\infty$. It is clear that $\bar{f}_m$ is convex and piecewise linear and satisfies (2.8).

Using the cuts we can reformulate (2.5)-(2.7) in a more explicit fashion. Let us in introduce aggregate vectors and matrices: $p_n = (p_{mn})_{m\in S(n)}$, $v_n = (v_{mn})_{m\in S(n)}$, $a_n = (a_{mj})_{m\in S(n),j\in J_m}$, $G_n = (g_{mj})_{m\in S(n),j\in J_m}$. With this notation (2.5)-(2.7) can be equivalently formulated as follows:

$$\textit{minimize} \;\; \eta_n = \tfrac{1}{2}\|x_n - \xi_n\|^2 + c^*_n x_n + p^*_n v_n \tag{3.3}$$

$$a_n + G^*_n x_n - E^*_n v_n \leq 0, \tag{3.4}$$

$$H_n x_n = b_n - D_n x_{\pi(n)}. \tag{3.5}$$

Here E_n is a zero-one matrix, whose j-th column has 1 at position l if the j-th cut in (3.4) is an objective cut for the l-th in order function $f_m(\cdot)$. The columns corresponding to feasibility cuts are zero. For simplicity we include direct constraints (2.7) into (3.4) as feasibility cuts.

We assume that there is at least one cut for each $f_m(\cdot)$, $m \in S(n)$, among (3.4), so that E_n has full row rank.

To describe the way in which cuts for the predecessor can be generated let us fix our attention on a specific class of methods for solving (3.3)-(3.5): the *active set methods* which proved useful for linear quadratic problems of similar structure (cf. [11,17,18]).

The main idea of active set methods is to choose a subset of linearly independent constraints from (3.4)-(3.5), solve the equality constrained subproblem obtained and revise the active set if optimality conditions for the whole problem are not satisfied. Each active set defines some submatrices G, E, H, D of G_n, E_n, H_n, D_n and subvectors a, b of a_n, b_n, which are used in equality constraints:

$$G^* x_n - E^* v_n = -a, \tag{3.6}$$

$$H x_n = b - D x_{\pi(n)}. \tag{3.7}$$

The necessary and sufficient conditions of optimality for (3.3),(3.6), (3.7) have now the form:

$$E\lambda = p_n,$$

$$x_n + G\lambda + H^*\mu = \xi_n - c_n.$$

We can always choose an active set so that E is of full row rank and $[\begin{matrix} E \\ G \end{matrix} \; H^*]$ is of full column rank. There can be many specific ways in which the active set can be altered [11,17,18], but there are always only two possible situations in which the method terminates: optimality with $\lambda \geq 0$ and (x_n, v_n) satisfying (3.4) and (3.5), or inconsistency of the active cuts with a certain inactive cut. These two cases determine the type of information that can be passed to the predecessing problem.

Lemma 1 *Let (3.3)-(3.5) be solvable for some $x_{\pi(n)}$ with the final active set (3.6)-(3.7). If the system of equations*

$$E\lambda = p_n, \tag{3.8}$$

$$G\lambda + H^*\mu = -c_n, \tag{3.9}$$

has a solution (λ, μ) with $\lambda \geq 0$, then

$$f_n(x_{\pi(n)}) \geq g^* x_{\pi(n)} + \alpha \;\; \textit{for all} \;\; x_{\pi(n)}, \tag{3.10}$$

where

$$g = D^*\mu, \tag{3.11}$$

$$\alpha = a^*\lambda - b^*\mu. \tag{3.12}$$

Proof. Consider the linear problem

$$\textit{minimize} \;\; l_n = c_n^* x_n + p_n^* v_n \tag{3.13}$$

$$G^* x_n - E^* v_n \leq -a, \tag{3.14}$$

$$H x_n = b - D x_{\pi(n)}. \tag{3.15}$$

It is a relaxation of (2.4), so the optimal value satisfies for each $x_{\pi(n)}$ the inequality

$$\hat{l}_n(x_{\pi(n)}) \leq f_n(x_{\pi(n)}).$$

On the other hand $\lambda \geq 0$ and μ satisfying (3.8)-(3.9) form a feasible dual solution to (3.13)-(3.15). Thus for each $x_{\pi(n)}$

$$\hat{l}_n(x_{\pi(n)}) \geq \lambda^* a + \mu^*(Dx_{\pi(n)} - b).$$

Combining the last two inequalities we obtain the required result.

Lemma 2 *For a given set of inequalities (3.4)-(3.5) the number of different objective cuts (3.10)-(3.12) is finite.*

Proof. Each cut (3.10)-(3.12), if it exists, is uniquely defined by the active set, and there can be only finitely many different active sets.

Lemma 3 *If the solution x_n to (3.3)-(3.5) at $x^0_{\pi(n)}$ is equal to ξ_n, then the cut (3.10)-(3.12) exists and supports the epigraph of the function*

$$\tilde{f}_n(x_{\pi(n)}) = min\{c_n^* x_n + \sum_{m \in S(n)} p_{mn} \tilde{f}_m(x_n) \mid H_n x_n = b_n - D_n x_{\pi(n)},\ x_n \geq 0\}$$

at $(x^0_{\pi(n)}, \tilde{f}_n(x^0_{\pi(n)}))$.

Proof. At $x_n = \xi_n$ the necessary and sufficient conditions of optimality for (3.13)-(3.15) and (3.3)-(3.5) are identical, so the cut must exist. Next, the constraints not included into the active set are satisfied at (x_n, v_n). Therefore $\hat{l}_n(x^0_{\pi(n)}) = \tilde{f}_n(x^0_{\pi(n)})$. Since (g, α) supports $\hat{l}_n(\cdot)$ at $x^0_{\pi(n)}$ it supports $\tilde{f}_n(\cdot)$ at $x^0_{\pi(n)}$, too.

Lemma 4 *Suppose that (3.4)-(3.5) are inconsistent for some $x^0_{\pi(n)}$. Then there exists an active set (3.6)-(3.7) such that one of the following conditions holds.*

(i) *There is a feasibility cut $\alpha + g^* x_n \leq 0$ among (3.4) and multipliers $\lambda \geq 0$ and μ such that*

$$E\lambda = 0, \tag{3.16}$$

$$g + G\lambda + H^*\mu = 0, \tag{3.17}$$

$$\alpha + \lambda^* a + \mu^*(Dx^0_{\pi(n)} - b) > 0. \tag{3.18}$$

(ii) *There is an equation $hx_n = \beta - dx_{\pi(n)}$ among (3.5) and multipliers $\lambda \geq 0$, μ and $\epsilon = \pm 1$ such that*

$$E\lambda = 0, \tag{3.19}$$

$$\epsilon h + G\lambda + H^*\mu = 0, \tag{3.20}$$

$$\epsilon((dx^0_{\pi(n)} - \beta) + \lambda^* a + \mu^*(Dx^0_{\pi(n)} - b)) > 0. \tag{3.21}$$

Proof. Suppose that the cut $\alpha + g^* x_n \leq 0$ is violated at the solution of the equality constrained subproblem and cannot be introduced into the active set. Then (3.16)-(3.18) with $\lambda \geq 0$ follow from [14, thm. 22.1]. If an equality constraint $hx_n = \beta - dx_{\pi(n)}$ is inconsistent with active cuts, in a similar way we get (3.19)-(3.21).

Using lemma 4 we can obtain cuts which must be satisfied by any $x_{\pi(n)}$. If case (i) holds, multiplying (3.14) by λ^* and adding (3.15) multiplied by μ^* we see that

$$\lambda^* G^* x_n + \mu^* H x_n - \lambda^* E^* v_n \leq -\lambda^* a + \mu^*(b - Dx_{\pi(n)}),$$

and, since x_n must satisfy $\alpha + g^* x_n \leq 0$,

$$\alpha + \lambda^* a + \mu^* (D x_{\pi(n)} - b) \leq 0.$$

In case (ii) in a similar fashion we obtain the cut

$$\epsilon \lambda^* a + \epsilon (d x_{\pi(n)} - \beta) + \epsilon \mu^* (D x_{\pi(n)} - b) \leq 0.$$

The new cut is violated at $x^0_{\pi(n)}$. These two cases can be put in one format

$$\bar{\lambda}^* a_n + \bar{\mu}^* (D_n x_{\pi(n)} - b_n) \leq 0, \tag{3.22}$$

by assigning zero multipliers to inactive cuts, multiplier 1 to the violated cut, and changing signs of (λ, μ) if $\epsilon = -1$. We can summarize it in the following lemma.

Lemma 5 *At any $x^0_{\pi(n)}$ for which (3.4)-(3.5) are inconsistent we can construct by (3.31) or (3.32) a feasibility cut*

$$\bar{\alpha} + \bar{g}^* x_{\pi(n)} \leq 0, \tag{3.23}$$

$$\bar{g} = D_n^* \bar{\mu}, \tag{3.24}$$

$$\bar{\alpha} = a_n^* \bar{\lambda} - b_n^* \bar{\mu}. \tag{3.25}$$

The number of such cuts possible is finite and they fully describe the set of $x_{\pi(n)}$ for which (3.4)-(3.5) are consistent.

Proof. Formulae (3.23)-(3.25) follow directly from (3.22). Each such cut is defined uniquely by the active set and the violated constraint, because (3.16)-(3.17) or (3.19)-(3.20) define uniquely (λ, μ) by the full column rank of $\begin{bmatrix} E & \\ G & H^* \end{bmatrix}$. The number of possible active sets for (3.4)-(3.5) is finite and for each active set there can be only finitely many violated constraints. Therefore, one can generate only only finitely many cuts (3.23)-(3.25). If $x^0_{\pi(n)}$ satisfies them, then it must satisfy (3.4)-(3.5), since otherwise we would be able to construct a new cut by lemma 4. The proof is complete.

For the linear problem (2.9)-(2.11) the cuts simplify slightly: there are no terms $a^* \lambda$ and $a_n^* \bar{\lambda}$ in (3.12) and (3.25).

4 Tasks

As we mentioned in section 2, our method for solving (2.1)-(2.3) consists of a number of tasks which can be executed in parallel and can exchange information in an asynchronous manner. With each node n of the tree $\mathcal{T}$ we asociate a task $SUB(n)$ whose function is to solve the regularized subproblem (3.3)-(3.5) corresponding to node n. The task $SUB(n)$ communicates with other tasks through two channels: $BOX(n)$ and $PIPE(n)$. Let us describe the channels and the tasks in more detail.

BOX(n)

In $BOX(n)$ the last solution x_n of (3.3)-(3.5) is stored. Only $SUB(n)$ may change its contents by overwriting x_n. The tasks $SUB(m)$ for $m \in S(n)$ may read x_n without destroying it. If $BOX(n)$ is empty and $SUB(m)$ attempts to read x_n, $SUB(m)$ waits until there will be new information available.

PIPE(n)

Through *PIPE*(n) cuts generated by the tasks *SUB*(m), $m \in S(n)$ are transmitted to *SUB*(n). *PIPE*(n) has a finite capacity which allows for storing at least one cut. When *SUB*(n) takes a cut from *PIPE*(n), the cut is deleted and new space in *PIPE*(n) is created. If *PIPE*(n) is full, the tasks (*SUB*(m), $m \in S(n)$ which attempt to put cuts to *PIPE*(n), wait until room for the next cut will be available.

The tasks *SUB*(n) have three different forms: for the root node, for the leaves $n \in \mathcal{N}_T$ and for the intermediate nodes. *SUB*(n) operates in two modes: 'go' and 'optimal' and updates the solution of (3.3)-(3.5) each time new information is available in the buffers. To simplify our description we asume that at the beginning every *SUB*(n), $n \notin \mathcal{N}_T$, has at least one objective cut for each $f_m(\cdot)$, $m \in S(n)$. The tasks start in mode 'go' and execute the following algorithm.

SUB(n) *for* $n \neq 1$ *and* $n \notin \mathcal{N}_T$

Step 1. Read $x_{\pi(n)}$ from *BOX*$(\pi(n))$.

Step 2. Get a cut from *PIPE*(n).

Step 3. If $x_{\pi(n)}$ did not change and *PIPE*(n) was empty, go to Step 4; otherwise set mode to 'go' and go to Step 5.

Step 4. If mode='optimal' go to Step 1; otherwise go to Step 8.

Step 5. Solve the subproblem (3.3)-(3.5) and delete from (3.4) the cuts that were inactive at the solution. If (3.3)-(3.5) was infeasible then go to Step 6. If (3.3)-(3.5) was solvable then go to Step 7.

Step 6. Clear *BOX*(n), generate the feasibility cut (3.22), put it into *PIPE*$(\pi(n))$ and go to Step 1.

Step 7. Write x_n into *BOX*(n) and generate the objective cut (3.10)-(3.12), if possible. If the objective cut exists then put it into *PIPE*$(\pi(n))$. Go to Step 1.

Step 8. If the tasks *SUB*(m) for all $m \in S(n)$ read the last x_n from *BOX*(n) and are in mode 'optimal', then go to Step 9; otherwise go to Step 1.

Step 9. If $x_n \neq \xi_n$ then set $\xi_n \Leftarrow x_n$ and go to Step 5; otherwise change mode to 'optimal' and go to Step 1.

Before proceeding to the other cases let us briefly comment on the above algorithm. There are two external sources of changes in the solution of (3.3)-(3.5): changes in $x_{\pi(n)}$ and new cuts. Only if these possibilities are exploited and no new cuts can be expected, because the sons are in mode 'optimal' (Step 8), we update the regularizing point ξ_n. If this is exploited too, we change the mode to 'optimal' to let our predecessor know that nothing new can be expected from us.

SUB(n) processes many cuts and most of them become soon outdated. However, owing to the deletion rule of Step 5, the size of (3.3)-(3.5) is bounded. The set of cuts that are stored (the *committee*) never has more than $m_x + |S(n)| + 1$ members: no more than $m_x + |S(n)|$ active cuts and one new cut read from *PIPE*(n). A specialized algorithm for updating the solution of (3.3)-(3.5) when a new cut is added has been developed in [17,18].

The task for leaves is much simpler: there are no cuts to process and the problem is linear.

$SUB(n)$ *for* $n \in \mathcal{N}_T$

Step 1. Read $x_{\pi(n)}$ from $BOX(\pi(n))$.

Step 2. If $x_{\pi(n)}$ is different from the last $x_{\pi(n)}$ set mode to 'go' and go to Step 3; otherwise set mode to 'optimal' and go to Step 1.

Step 3. Solve the subproblem (2.9)-(2.11). If (2.9)-(2.11) was solvable then go to Step 4; otherwise go to Step 5.

Step 4. Generate the objective cut (3.10)-(3.12), put it into $PIPE(\pi(n))$ and go to Step 1.

Step 5. Generate the feasibility cut (3.22), put it into $PIPE(\pi(n))$ and go to Step 1.

The root task is responsible for detecting optimality or infeasibility and terminating the whole method.

$SUB(1)$

Step 1. Get a cut from $PIPE(1)$. If $PIPE(1)$ is empty then go to Step 4; otherwise go to Step 2.

Step 2. Solve the subproblem (3.3)-(3.5) and delete from (3.4) the cuts that were inactive at the solution. If (3.3)-(3.5) was infeasible then go to Step 7. If (3.3)-(3.5) was solvable then go to Step 3.

Step 3. Write x_1 into $BOX(1)$ and go to Step 1.

Step 4. If the tasks $SUB(m)$ for all $m \in \mathcal{S}(1)$ read the last x_1 from $BOX(1)$ and are in mode 'optimal', then go to Step 5; otherwise go to Step 1.

Step 5. If $x_1 \neq \xi_1$ then set $\xi_1 \Leftarrow x_1$ and go to Step 2; otherwise go to Step 6.

Step 6. Terminate (*optimal solution found*).

Step 7. Terminate (*the problem is infeasible*).

If $SUB(1)$ terminates, all other tasks terminate, too; their last solutions contain then the solution to the original problem.

5 Convergence

Out aim in this section is to prove that the method after a finite time either discovers inconsistency in the problem or finds its optimal solution (recall that we assume througout this paper that the problem is bounded). We shall use τ to denote time that passed from the start of the method.

To avoid deadlocks and races we shall need two additional assumptions.

(A1) If a new x_n is written into $BOX(n)$, then after a finite time each $SUB(m)$, $m \in \mathcal{S}(n)$ will get acess to $BOX(n)$.

(A2) If $SUB(m)$ for $m \in \mathcal{S}(n)$ reads x_n from $BOX(n)$, then the mode of $SUB(m)$ is changed to 'go' before $SUB(n)$ checks it at Step 8.

Let us introduce two notions concerning asymptotic behavior of our subproblems.

Definition 1 *We say that $SUB(n)$ for $n \neq 1$ is* stable from above *if there exists a finite time τ_n such that the contents of $BOX(\pi(n))$ does not change for $\tau \geq \tau_n$. The task $SUB(1)$ is* stable from above *if it is feasible for all $\tau \geq 0$.*

Definition 2 *We say that $SUB(n)$ is* terminally optimal *if there exists a finite time $\hat{\tau}_n$ such that $SUB(n)$ stays in mode 'optimal' for all $\tau \geq \hat{\tau}_n$.*

We are now ready to carry out our analysis.

Lemma 6 *Suppose that $SUB(n)$ is in mode 'optimal' at time τ. Then the tasks $SUB(m)$ for $m \in \mathcal{S}(n)$ ar in mode 'optimal' at time τ.*

Proof. Out assertion is true for leaves $n \in \mathcal{N}_T$. Suppose that it is true for all $m \in \mathcal{S}(n)$. We shall prove it for n. Let $SUB(n)$ be in mode 'optimal' at time τ. Then at some time $\tau_n \leq \tau$ $SUB(n)$ entered Step 8 and the tasks $SUB(m)$, $m \in \mathcal{S}(n)$ were at mode 'optimal' at time instants $\tau_m \in [\tau_n, \tau]$. Each $SUB(m)$ can change its mode only after receiving a new x_n from $BOX(n)$ or a new cut from $PIPE(m)$. In the interval $[\tau_m, \tau]$ the solution x_n does not change, because $SUB(n)$ stays in mode 'optimal'. Next, by our inductive assumption $SUB(j)$, $j \in \mathcal{S}(m)$ are in mode 'optimal', so $PIPE(m)$ remains empty. Consequently, $SUB(m)$, $m \in \mathcal{S}(n)$ stay in mode 'optimal' in the intervals $[\tau_m, \tau]$.

Lemma 7 *Suppose that $SUB(n)$ is in mode 'optimal' at time τ. Then x_n solves the linear problem (2.4).*

Proof. Out assertion is true for leaves $n \in \mathcal{N}_T$. Suppose that it is true for all $m \in \mathcal{S}(n)$. We shall prove it for n. Let $\tau_n \leq \tau$ be the last time at which x_n changed. By lemma 6, all $SUB(m)$, $m \in \mathcal{S}(n)$ are in mode 'optimal' at time τ. On the other hand, by Step 3 each $SUB(m)$ changed its mode to 'go' at a certain $\tau_m \in [\tau_n, \tau]$. So, each $SUB(m)$, $m \in \mathcal{S}(n)$ executed at least once Step 5 in the time interval $[\tau_m, \tau]$. Let $\hat{\tau}_m$ be the last time in this interval at which Step 5 was executed by $SUB(m)$. Since $SUB(m)$ is in mode 'optimal' at τ we must have had $x_m = \xi_m$ at $\hat{\tau}_m$. By our inductive assumption and lemma 3, the last objective cut generated by $SUB(m)$ supported $f_m(\cdot)$ at x_n. The cut was read by $SUB(n)$ in the interval $[\hat{\tau}_m, \tau]$ by virtue of (A1)-(A2). But x_n did not change in $[\hat{\tau}_m, \tau]$; hence $\bar{f}_m(x_n) \geq f_m(x_n)$. Since $\bar{f}_m \leq f_m$ we obtain $\bar{f}_m(x_n) = f_m(x_n)$ for all $m \in \mathcal{S}(n)$. Consequently, x_n solves (2.4). If $m \in \mathcal{S}(n)$ is a leaf, the analysis is simpler, because each objective cut is then a supporting cut.

Lemma 8 *There are finitely many possible committees for each $SUB(n)$.*

Proof. Our assertion is trivial for leaves $n \in \mathcal{N}_T$. Suppose that it is true for all $m \in \mathcal{S}(n)$. We shall prove it for n. Each committee is a set of cuts generated by the tasks $SUB(m)$, $m \in \mathcal{S}(n)$. By our inductive assumption each successor of n may have only finitely many committees. By lemmas 2 and 5 each committee may define only finitely many cuts. Therefore only finitely many committees for $SUB(n)$ can be formed from these cuts.

Lemma 9 *Suppose that $SUB(n)$ is stable from above. Then ξ_n is changed only finitely many times.*

Proof. Let $x_{\pi(n)}$ be fixed for $\tau \geq \tau^0$ and let ξ_n be changed at time instants $\tau^k \geq \tau^0$, $k = 1, 2, \ldots$. Let x_n^k denote the solution to (3.3)-(3.5) at τ^k. By Step 11, the regularizing point in the interval $[\tau^k, \tau^{k+1}]$ is given by $\xi_n^{k+1} = x_n^k$. It is changed at τ^{k+1}, so $x_n^{k+1} \neq \xi_n^{k+1}$. Let

$$\hat{\tau}^k = \inf\{\tau \in [\tau^k, \tau^{k+1}] : \; x_n(\tau) = x_n^{k+1}\}.$$

Consider $SUB(m)$, $m \in S(n)$. By Step 8 of $SUB(n)$, each $SUB(m)$, $m \in S(n)$ reads x_n^{k+1} at some time instant $\tau_m^k \geq \tau^k$, changes mode to 'go', and reaches mode 'optimal' at some time instant $\sigma_m^k \leq \tau^{k+1}$. By (A2), $\tau_m^k < \sigma_m^k$. Therefore, to reach optimality at σ_m^k, $SUB(m)$ must execute Step 5 with $x_m = \xi_m$ in the interval $[\tau_m^k, \sigma_m^k]$. By lemma 3 the objective cut at this point supports $f_m(\cdot)$ at x_n^{k+1}, i.e. $\bar{f}_m(x_n^{k+1}) = f_m(x_n^{k+1})$. Summing up, at each time instant τ^k the following relations hold

(i) x_n^{k+1} solves (2.5)-(2.7) with $\xi_n = \xi_n^k$;

(ii) $\bar{f}_m(x_n^{k+1}) = f_m(x_n^{k+1})$ for all $m \in S(n)$.

These two conditions imply that at τ^k an *exact serious step* of the regularized decomposition method of [17] for solving the problem

$$\begin{gathered} \textit{minimize} \;\; F_n(x_n) = c_n^* x_n + \sum_{m \in S(n)} p_{mn} f_m(x_m) \\ H_n x_n = b_n - D_n x_{\pi(n)}, \\ x_n \geq 0 \end{gathered}$$

is executed ($x_{\pi(n)}$ is fixed). It follows from the theory developed in [17] that after finitely many such steps the minimum of F_n will be reached and no more steps will be possible.

Lemma 10 *If $SUB(n)$ is stable from above then all its successors $SUB(m)$ for $m \in S(n)$ are stable from above.*

Proof. By lemma 9, ξ_n can be changed only finitely many times. Hence there is τ_0 such that for $\tau \geq \tau_0$ both $x_{\pi(n)}$ and ξ_n remain constant. The solution x_n to (3.3)-(3.5) does not change when inactive cuts are deleted. It is unique for a given committee, owing to the existence of the quadratic regularizing term in (3.3). Consequently, x_n may change only by introduction of a cut which cuts-off the previous solution. In this case the minimum value of (3.3)-(3.5) increases. By lemma 8 there can be only finitely many different committees at $SUB(n)$, which implies that x_n may be changed only finitely many times. The proof is complete.

Lemma 11 *If $SUB(n)$ is stable from above then it is terminally optimal.*

Proof. Our assertion is obvious for leaves $n \in \mathcal{N}_T$. Suppose that it is true for all $m \in S(n)$. We shall prove it for n. By lemma 10, the successors $SUB(m)$, $m \in S(n)$ are stable from above. By our inductive assumption they are terminally optimal. Let τ_0 be such a time instant that for $\tau \geq \tau_0$ $x_{\pi(n)}$ and x_n do not change and $SUB(m)$, $m \in S(n)$ are in mode 'optimal'. If $SUB(n)$ were in mode 'go' at some time $\tau \geq \tau_0$ it would have to enter Step 8. But x_n does not change for $\tau \geq \tau_0$, so we would have $x_n = \xi_n$, mode would be set to 'optimal' and $SUB(n)$ would start infinite cycling between Steps 1 and 4.

It is now easy to prove our main result.

Theorem 1 *After a finite time the method either discovers inconsistency in the problem and stops at Step 6 of $SUB(1)$ or finds an optimal solution and stops at Step 5 of $SUB(1)$. In the latter case the solution is given by ξ_n, $n \in \mathcal{N}$.*

Proof. Suppose that $SUB(1)$ is not stable from above. Then after a finite time it stops at Step 6 with inconsistent feasibility cuts. The cuts approximate the domain of $f_1(\cdot)$ from outside, so the problem is infeasible in this case. Suppose now that $SUB(1)$ is stable from above. By lemma 11 it is terminally optimal and after a finite time it stops at Step 6. Then by lemmas 6 and 7 all tasks are in mode 'optimal' with $x_n = \xi_n$, $n \in \mathcal{N}$ solving the corresponding problems (2.4). The proof is complete.

6 Conclusions

Our decomposition approach differs from earlier methods in two ways.

- It has regularizing quadratic terms in all subproblems (except for the leaves) which stabilize their solutions and allow for deletion of inactive cuts.
- All subproblems are solved in parallel and exchange information in an asynchronous manner. This speeds up the flow of information between stages.

In spite of these modifications, the method shares the finite termination property of classical approaches.

References

[1] D.P. Bertsekas, Distributed dynamic programming, *IEEE Transactions on Automatic Control* AC-27(1982) 610-616.

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

[3] J. Bisschop and A. Meeraus, Matrix augmentation and partitioning in in the updating of the basis inverse, *Mathematical Programming* 30(1984) 71-87.

[4] G.B. Dantzig and A. Madansky, On the solution of two-stage linear programs under uncertainty, in *Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability, vol I*, University of California Press, Berkeley 1961, pp. 165-176.

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

[6] R. Fourer, Solving staircase linear programs by the simplex method, 1: inversion, *Mathematical Programming* 23(1982) 274-313.

[7] R. Fourer, Solving staircase linear programs by the simplex method, 2: pricing, *Mathematical Programming* 25(1983) 251-292.

[8] J.K. Ho, T.C. Lee and R.P. Sundarraj, Decomposition of linear programs using parallel computation, technical report, College of Business Administration, University of Tennessee, Knoxville, 1987.

[9] J.K. Ho and A.S. Manne, Nested decomposition for dynamic models, *Mathematical Programming* 6(1974) 121-140.

[10] P, Kall, Computational methods for solving two-stage stochastic linear programming problems, *ZAMT* 30(1979) 261-271.

[11] K. C. Kiwiel, A dual method for certain positive semidefinite quadratic programming problems, technical report, Systems Research Institute, Warsaw 1987.

[12] B. Murtagh, *Advanced Linear Programming*, McGraw-Hill, 1981.

[13] A. Propoi and V. Krivonozhko, The simplex method for dynamic linear programs, RR-78-14, IIASA, Laxenburg, 1978.

[14] R.T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton 1970.

[15] R.T. Rockafellar and R.J.-B. Wets, A Lagrangian finite generation technique for solving linear quadratic problems in stochastic programming, *Mathematical Programming Study* 28(1986) 63-93.

[16] R.T. Rockafellar and R.J.-B. Wets, Scenarios and policy aggregation in optimization under uncertainty, WP-87-119, IIASA, Laxenburg 1987.

[17] A. Ruszczyński, A regularized decomposition method for minimizing a sum of polyhedral functions, *Mathematical Programming* 35(1986) 309-333.

[18] A. Ruszczyński, Regularized decomposition of stochastic programs: algorithmic techniques and numerical results, *Operations Research* , to appear.

[19] A.Ruszczyński, Modern techniques for linear dynamic and stochastic programs, in: *Theory, software and testing examples for decision support systems*, A. Lewandowski and A. Wierzbicki (eds.), WP-87-26, IIASA, Laxenburg 1987, pp.27-43.

[20] A. Ruszczyński, Regularized decomposition and augmented Lagrangian decomposition for angular linear programming problems,WP-88-88, IIASA, Laxenburg 1988.

[21] B. Strazicky, Some results concerning an algorithm for the discrete recourse problem, in: *Stochastic Programming*, M. Dempster (ed.), Academic Press, London 1980, pp. 263-274.

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

[23] R. J.-B. Wets, Large scale linear programming techniques in stochastic programming, in: *Numerical Methods in Stochastic Programming*, Y. Ermoliev and R. Wets (eds), Springer-Verlag, Berlin 1986 (to appear).

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