A splitting method for stochastic programs*

Teemu Pennanen†
Markku Kallio‡
Department of Economics and Management Science
Helsinki School of Economics
PL 1210
00101 Helsinki, Finland
January 8, 2002

Abstract

This paper derives a new splitting-based decomposition algorithm for convex stochastic programs. It combines certain attractive features of the progressive hedging algorithm of Rockafellar and Wets, the dynamic splitting algorithm of Slinger and Rockafellar and an algorithm of Korf. We give two derivations of our algorithm. The first one is very simple, and the second one yields a preconditioner that can be used to speed up the convergence.

1 Introduction

Consider a stochastic program over a scenario tree of a finite set $I$ of nodes. Except for the initial node, which is denoted by $0$, every node $i \in I$ has a unique predecessor node that will be denoted by $i_\prec$. The probability of reaching node $i$ will be denoted by $\pi_i$, and it is assumed strictly positive. Every node $i \in I$ has an associated decision variable $x_i \in \mathbb{R}^{n_i}$, and the problem is to

$$\text{minimize} \quad \sum_{i \in I} \pi_i f_i(x_i),$$
$$\text{subject to} \quad A_i x_i + B_i x_{i_\prec} = b_i \quad \forall i \in I \setminus \{0\},$$

(SP)

where $f_i$ are closed convex functions taking values in $(-\infty, +\infty]$, $A_i \in \mathbb{R}^{m_i \times n_i}$, $B_i \in \mathbb{R}^{m_i \times n_{i_\prec}}$, and $b_i \in \mathbb{R}^{m_i}$. The value $f_i(x_i)$ can be interpreted as the penalty that has to be paid if node $i$ is reached. We emphasize that by allowing infinite penalties, any constraints of the form

$$x_i \in X_i,$$

where $X_i$ is a closed convex set, can be incorporated into $f_i$.

Stochastic programs are typically very large due to the large number of scenarios needed to represent the underlying uncertainty accurately enough. Several decomposition algorithms have been developed to take advantage of the special structure of the problem; see the survey articles of Birge [2] and Ruszczynski [15]. The purpose of this paper is to derive yet another decomposition

---

*This work was supported by the Finnish Foundation for Economic Education under grant number 20728, and by Jenny ja Antti Wihuri Foundation.

†pennanen@hkk.fi
‡kallio@hkk.fi
algorithm for (SP), using operator splitting methods. The algorithm achieves a decomposition over the nodes of the scenario tree, and the subproblems are coordinated by solving simplified stochastic programs where the objective is replaced by a quadratic function. These “master problems” can be solved with standard methods of linear algebra, and it turns out that only one matrix factorization is needed in solving all of them in the course of the algorithm. The algorithm is closely related to [10, Algorithm 17], but the approach we take here is simpler and applies to a more general problem format. Another contribution of this paper is a new scaling strategy that is suggested by well known properties of the proximal point algorithm [13] from which our decomposition algorithm is derived.

Our algorithm has close connections with the progressive hedging algorithm of Rockafellar and Wets [14], the dynamic splitting algorithm of Salinger and Rockafellar [16, 17], and the decomposition algorithm of Korf [8]. Like progressive hedging, our algorithm can be easily derived from the Douglas-Rachford operator splitting algorithm. The main difference comes from the fact that progressive hedging is based on scenario-wise formulation with nonanticipativity constraints, which leads to decomposition over the scenarios, and to a very simple master problem. Both the dynamic splitting and the algorithm of [8] were derived by applying a splitting algorithm to a primal-dual formulation of the original problem. Both algorithms decompose the original problem into nodewise subproblems and a linear coordination problem like our algorithm. The nodewise subproblems are variational inequalities that in [8] were solved directly by a pivoting algorithm, but in [16] dualization was used to reformulate them as minimization problems. The objectives in the resulting minimization problems involve certain conjugate functions whose evaluation is a nontrivial task in general. The algorithm derived in this paper combines the simplicity of the progressive hedging with the nodewise decomposition of dynamic splitting and the algorithm of [8].

In the next section, we derive the decomposition algorithm from the Douglas-Rachford splitting algorithm. In Section 3, we give an alternative, a bit more complicated derivation based on the method of partial inverses of Spingarn [38], and obtain a way to scale the problem. In the last section, we present the results of our first computational experiments.

2 Douglas-Rachford splitting — a decomposition algorithm

Consider the product space $X = \prod_{i \in I} \mathbb{R}^{n_i}$ equipped with the inner product

$$\langle x, y \rangle = \sum_{i \in I} \pi_i x_i \cdot y_i$$

and the corresponding norm $\|x\| = \left(\sum_{i \in I} \pi_i \|x_i\|^2\right)^{1/2}$. Let $C \subset X$ denote the affine set described by the dynamic constraints in (SP). Defining the function $f$ on $X$ by $f(x) = \sum_{i \in I} \pi_i f_i(x_i)$, problem (SP) can be written as

$$\text{minimize } f(x) + \delta_C(x),$$

where $\delta_C$ is the indicator function of $C$. By the sum-rule of subdifferentiation, we have $\partial(f + \delta_C) = \partial f + \partial \delta_C$, where equality holds if $0 \in \text{ri}(\text{dom } f - C)$ [12, Theorem 23.8]. Thus, the inclusion

$$T_1(x) + T_2(x) \ni 0, \quad (1)$$

where $T_1 = \partial f$ and $T_2 = \partial \delta_C$, is a sufficient condition for optimality in (SP), and under the constraint qualification $0 \in \text{ri}(\text{dom } f - C)$ it is also necessary.

Several operator splitting algorithms have been developed for problems of the form (1); see for example [4, 6]. In particular, we can use the Douglas-Rachford splitting algorithm which can be written as follows.

Douglas-Rachford splitting
1. Compute
\[ z^{k+1} = (I + T_1)^{-1}(x^k + y^k); \]
2. Compute
\[ x^{k+1} = (I + T_2)^{-1}(z^{k+1} - y^k), \]
and set \( y^{k+1} = y^k - z^{k+1} + x^{k+1}. \)

In our case, step 1 means that \( z^{k+1} \) solves the inclusion
\[ \partial f(z) + z \ni x^k + y^k, \]
or equivalently, minimizes the function
\[ f(z) + \frac{1}{2} \| z - (x^k + y^k) \|^2_n. \]

Similarly, step 2 means that \( x^{k+1} \) minimizes
\[ \delta_C(x) + \frac{1}{2} \| x - (z^{k+1} - y^k) \|^2_n, \]
or in other words, \( x^{k+1} \) is the projection of \( z^{k+1} - y^k \) on the feasible set \( C \). It follows that if \( x^k \in C \) and \( y^k \in C^\perp \), then \( x^k \in C \) and \( y^k \in C^\perp \) for all \( k \), so in particular, \( (I + T_2)^{-1}(z^{k+1} - y^k) = (I + T_2)^{-1}(z^{k+1}) \). Using the definitions of \( f \), \( C \) and \( \| \cdot \|_n \), we can write the Douglas-Rachford splitting algorithm as follows.

**Algorithm 1**

0. Choose \( x^0 \in C \) and \( y^0 \in C^\perp; \)
1. For each \( i \in I \)
   
   minimize \[ f_i(z_i) + \frac{1}{2} \| z_i - (x_i^k + y_i^k) \|^2, \]
   
   for \( z_i^{k+1} ; \)
2. \[ \begin{align*}
   \text{minimize} & \quad \sum_{i \in I} \frac{\alpha_i}{2} \| x_i - z_i^{k+1} \|^2 \\
   \text{subject to} & \quad Ax_i + B_i x_i = b_k \quad \forall i \in I \setminus \{0\} \\
   \text{for} & \quad x^{k+1}, \text{ set} \\
   & \quad y^{k+1} = y^k - z^{k+1} + x^{k+1}, \\
   \end{align*} \]
   \( k = k + 1 \) and go to step 1.

Note that the minimization in step 1 has been split into a separate subproblem for each node of the scenario tree. The additional quadratic term guarantees that each one has a unique solution. All the dynamic constraints of the original problem are dealt with in step 2, where one is asked to solve a stochastic program which is obtained from the original one by replacing the nonlinear extended real-valued objective by a quadratic one. Such problems can be solved with well-known methods of linear algebra or by recursion as in [16, 17, 3, 19]. Viewing step 2 as a projection, suggests the use of matrix factorization techniques [7]. Since only the point to be projected changes from iteration to iteration, one can use the same factorization at every iteration.

The general convergence properties of the Douglas-Rachford algorithm yield the following; see [6, Proposition 6].

**Theorem 1** If \( f_i \) are closed proper convex functions, then \( \{x^k\} \) converges to a solution of \( (SP) \), provided one exists.
Note that [6, Proposition 6] allows inexact computations as well as over/under-relaxed updates. For simplicity of presentation, we will not consider these extensions explicitly.

It is interesting to compare Algorithm 1 with the progressive hedging algorithm of [14] (see also [11, 9]), which can be derived by applying the Douglas-Rachford algorithm to the "split-variable" (scenario-wise) formulation of (SP), much like we have done above; see Ruszczyński [15]. Progressive hedging algorithm also consist of a sequence of simple minimization problems and projection steps. In progressive hedging, the decomposition is achieved over the scenarios of the decision tree, as opposed to nodes as in Algorithm 1. On the other hand, in progressive hedging, the projection step comes down to computing conditional expectations, which is very easy to implement in the case of a finite scenario tree.

From the implementation point of view, Algorithm 1 is closer to the dynamic splitting algorithm developed in [16, 17]. There, as in Algorithm 1, the decomposition is achieved over the nodes, and the projection step involves the solution of a stochastic program with a quadratic objective. The main difference is that, in Algorithm 1, the objectives of the subproblems in step 1 can be directly obtained from the original problem, whereas in dynamic splitting, one has to compute certain conjugate functions, which can be difficult. Also, in the projection step of dynamic splitting, the quadratic term changes between the iterations, whereas in Algorithm 1, the coefficient matrix remains fixed throughout the algorithm.

In [8], Korf presented an algorithm that also decomposes a stochastic program over the nodes, but there, also the coordination step was decomposable, and instead of minimization problems, the nodewise subproblems were in the form of variational inequalities.

3 Method of partial inverses – scaling

In our first computational experiments we found that if Algorithm 1 is implemented as such, it converges slowly; see Section 4. It is well known that splitting methods are very sensitive to scaling of the problem, that is, on the choice of the inner product on the underlying space. The purpose of this section is to present an alternative derivation of Algorithm 1 that suggests a particular way to scale the problem.

Write the feasible set of (SP) as $C = S + a$, where $S$ is a subspace and $a$ is a feasible point. Setting $\bar{x} = x - a$ and $\bar{f}(x) = f(x + a)$, (SP) can be written as

$$\text{minimize} \quad \bar{f}(\bar{x}) + \delta_S(\bar{x}).$$

Just as in Section 2, we obtain the optimality condition $\partial \bar{f}(\bar{x}) + \partial \delta_S(\bar{x}) \ni 0$. Using the fact that $S$ is a subspace, it is easily verified that

$$\partial \delta_S(x) = \begin{cases} S^\perp & \text{if } x \in S, \\ \emptyset & \text{otherwise}, \end{cases}$$

where $S^\perp$ is the orthogonal complement of $S$. The optimality condition can thus be written as

$$x \in S, \quad y \in S^\perp, \quad y \in T(x), \quad (2)$$

where $T = \partial \bar{f}$.

Problem (2) is exactly in the form to which Spingarn’s method of partial inverses (MPI) can be applied [18]. Spingarn showed that (2) is equivalent to

$$T_S(z) \ni 0, \quad (3)$$

where $T_S$ is a monotone mapping called the partial inverse of $T$, and that the proximal point algorithm applied to (3) can be written in the following form.

Method of Partial Inverses
0. Choose $x^0 \in S$ and $y^0 \in S^+$;
1. Find $z^{k+1}, w^{k+1} \in H$ such that
\[
   x^k + y^k = z^{k+1} + w^{k+1}
\]
\[
   \frac{1}{c} P_{D} w^{k+1} + P_{S\perp} w^{k+1} \in T \left( P_{S} z^{k+1} + \frac{1}{c} P_{S\perp} z^{k+1} \right);
\]
2. Let $x^{k+1} = P_{S} z^{k+1}$ and $y^{k+1} = P_{S\perp} w^{k+1}, k = k + 1$ and go to 1,
where $c$ is the (in this case, constant) step-size parameter of the proximal point algorithm, and $P_D$ denotes the projection (with respect to the inner product $\langle \cdot, \cdot \rangle_2$) onto a set $D$.

If $c = 1$, step 1 can be written as $z^{k+1} = (I + T)^{-1}(x^k + y^k)$, and MPI becomes a special case of the Douglas-Rachford algorithm, as noted by Edstein [4]; see also Edstein and Berteskas [5]. This is how MPI is usually applied [18, 11]. It is well-known, however, that the proximal point algorithm tends to converge faster if the step-size parameter is kept high. We will now proceed with an arbitrary value of $c$, and reformulate MPI into an implementable form.

Defining $M_c = P_{S} + c^{-1} P_{S\perp}$, the inclusion in step 1 can be written as
\[
   \frac{1}{c} M_c^{-1} w^{k+1} \in T(M_c z^{k+1}) \iff \frac{1}{c} w^{k+1} \in (M_cTM_c)(z^{k+1}).
\]

Eliminating $w^{k+1}$ from the equations, MPI becomes
0. Choose $x^0 \in S$ and $y^0 \in S^+$;
1. Solve
\[
   (M_cTM_c)(z) + \frac{1}{c} (z - x^k - y^k) \ni 0
\]
for $z^{k+1};$
2. Set
\[
   x^{k+1} = P_{S} z^{k+1}, \\
   y^{k+1} = y^k - P_{S\perp} z^{k+1},
\]
$k = k + 1$, and go to 1.

Note that since $P_{S\perp} = I - P_S$, the update formula for $y^k$ can be written as $y^{k+1} = y^k - z^{k+1} + x^{k+1}$. Since $M_c$ is symmetric, we have by [12, Theorem 23.9] that $M_cTM_c = \partial(f + M_c)$, so step 1 means that $z^{k+1}$ minimizes the function
\[
   f(M_c z + a) + \frac{1}{2c} \| z - (x^k + y^k) \|^2_2.
\]

Making the change of variables
\[
   z := M_c z + a, \quad x := M_c x + a, \quad y := M_c y,
\]
we get the following.

0. Choose $x^0 \in C$ and $y^0 \in C^+$;
1. minimize $f(z) + \frac{1}{2c} \| M_c^{-1} (z - (x^k + y^k)) \|^2_2$
for $z^{k+1};$

5
2.  
\[
\text{minimize} \quad \frac{1}{2} \| M^{-1}_c (x - z^{k+1}) \|_H^2 \\
\text{subject to} \quad x \in C
\]
for \( x^{k+1} \), and set 
\[
y^{k+1} = y^k - z^{k+1} + x^{k+1}
\]
\( k = k + 1 \), and go to 1,

where \( C \) is the feasible set of \((SP)\). We see that this is just the Douglas-Rachford splitting algorithm applied to \((1)\) under the new inner product 
\[
\langle x, y \rangle_{H_c} = \langle H_c x, y \rangle,
\]
where \( H_c = M^{-1}_c \Pi M^{-1}_c \), and \( M^{-1}_c = (P_S + c^{-1}P_{g,\perp})^{-1} = P_S + cP_{g,\perp} \). Since \( P_S \) and \( P_{g,\perp} \) are projections with respect to the inner product \( \langle \cdot, \cdot \rangle_{\Pi} \), they commute with \( \Pi \), so that
\[
H_c = \Pi M^{-2}_c = \Pi (P_S + c^2 P_{g,\perp}). \quad (4)
\]
Noticing that \( P_{g,\perp} = I - P_S \), where \( S \) is the kernel of the constraint matrix in \((SP)\), the computation of \( H_c \) comes down to forming the projection matrix for \( S \); see [7].

Note that if \( c = 1 \), then \( H_c = I \) and we recover Algorithm 1. The fact that \( c \) is the step-size parameter of the proximal point algorithm, suggests choosing \( c > 1 \). The problem with this, however, is that the corresponding inner product need not be separable, and then we cannot decompose the minimization in step 1 over the nodes as in Algorithm 1. A natural idea then is to approximate the matrix \( H_c \) by a matrix \( D \) such that
\[
\langle y, Dx \rangle = \sum_{i \in I} y_i \cdot D_{ii} x_i.
\]
A very simple choice is to let \( D \) be the diagonal part of \( H_c \). While a high value of \( c \) corresponds to a large step-size in the proximal point algorithm, the (block-)diagonal approximation of \((4)\) becomes worse as \( c \) increases. It is thus a good idea to try different values \( c > 1 \) for a given problem.

4 Numerical experiments

Algorithm 1 was implemented for a stochastic program that arises in asset liability management of a Finnish pension insurance company [1]; see also [19]. This problem can be written as \((SP)\), where each \( f_i \) is a sum of exponential and quadratic terms. Six symmetric scenario trees with five stages were randomly generated by a time series model. The branching structures and the numbers of variables are shown in Table 1.

<table>
<thead>
<tr>
<th>problem</th>
<th>branching structure</th>
<th>variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 × 2 × 2 × 2 × 2</td>
<td>310</td>
</tr>
<tr>
<td>2</td>
<td>4 × 4 × 2 × 2 × 2</td>
<td>1170</td>
</tr>
<tr>
<td>3</td>
<td>8 × 2 × 2 × 2 × 2</td>
<td>2330</td>
</tr>
<tr>
<td>4</td>
<td>2 × 4 × 2 × 2 × 2</td>
<td>4650</td>
</tr>
<tr>
<td>5</td>
<td>16 × 4 × 2 × 2 × 2</td>
<td>8490</td>
</tr>
<tr>
<td>6</td>
<td>16 × 4 × 4 × 2 × 2</td>
<td>16810</td>
</tr>
</tbody>
</table>

Table 1: Test problems

We tried Algorithm 1 with and without the scaling strategy presented in the previous section. In all cases, we used the starting points \( x^0 = P c 0 \) and \( y^0 = 0 \), and the progress of the algorithm was measured by
\[
e_k = \| x^{k+1} - x^k \|_\infty.
\]
<table>
<thead>
<tr>
<th>problem</th>
<th>iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>117</td>
</tr>
<tr>
<td>2</td>
<td>140</td>
</tr>
<tr>
<td>3</td>
<td>154</td>
</tr>
<tr>
<td>4</td>
<td>378</td>
</tr>
<tr>
<td>5</td>
<td>521</td>
</tr>
<tr>
<td>6</td>
<td>441</td>
</tr>
</tbody>
</table>

Table 2: Number of iterations needed to reach the accuracy $e_k \leq 0.01$

which is motivated by the observation that if $e_k = 0$, then $x^k$ is optimal. Indeed, since $z^{k+1}$ satisfies

$$
\partial f(z^{k+1}) \ni y^k - (z^{k+1} - x^k),
$$

where $x^k \in C$ and $y^k \in C^-$, we have that if $e_k = 0$, then $z^{k+1}$ satisfies the optimality condition (1). The implementation was done in FORTRAN 77 programming language, and the subproblems in step 1 were solved with MINOS 5.4 as a subroutine.

Figure 1 shows the development of $\log(e_k)$ for problems 1–6 as a function of $k$. The bundle of six curves converging just below 1 in 1000 iterations resulted when just the scaling $D = \Pi$ (i.e. Algorithm 1 without the additional scaling of Section 3) was used. Next, the diagonal scaling $D = \Pi \text{diag}(M_c^{-1})$, suggested by (4), was employed. The parameter value of $c = 5.7$ was found by the method of trial and error on a small set of small problems. This resulted in the six remaining curves in Figure 1.

It can be seen that with the additional scaling, the size of the error reduces much faster for all problems than without the scaling. In the current implementation, the time to compute the diagonal scaling matrix took less than 5 percent of the total computation time, which seems reasonable in view of the improved convergence properties. With the error tolerance $e_k \leq 10^{-2}$, the iteration counts with scaling are shown in Table 2.

![Figure 1: $\log(e_k)$ as a function of $k$ for problems 1–6 with and without scaling](image-url)
Figure 1 clearly displays a feature that was observed in all the test runs: at first, the error reduces relatively fast but after some point, the algorithm seems to stall. Similar “fat tails” have been reported in [17]. The reason for the stalling is not clear, but nevertheless, the proposed scaling seems to reduce the effect.

It is clear that more tests and a more careful implementation of Algorithm 1 will required in order to draw definite conclusions about its practicality. It would also be interesting to study the causes of the sudden stalling that was observed in the tests. These are subjects of future research.

Acknowledgments. The authors would like to thank Jonathan Eckstein for suggesting several improvements to an earlier manuscript and for suggesting the use of Douglas-Rachford splitting that lead to the simple derivation of our algorithm in Section 2.

References


