A Memory-Efficient Implementation of Multi-Period Two- and Multi-Stage Stochastic Programming Models

Bruno A. Calfa*
bacalfa@gmail.com

April 14, 2014

Abstract

The objective of this paper is to describe a method of implementing multi-period two- and multi-stage Stochastic Programming (SP) models with exogenous uncertainty that is modeling-platform and programming-language independent. The proposed implementation approach generates an implicit extensive form of the SP model in contrast to an explicit formulation, which explicitly accounts for the sequence of decisions, thus introducing redundant variables and constraints in the model. The efficiency of the proposed implementation approach with respect to memory usage, thus problem size, is achieved with the introduction of three sets of auxiliary parameters in the mathematical formulation of the deterministic equivalent stochastic program. The three parameters capture the non-anticipativity condition, the mapping between scenarios and stages, and the structure of the scenario tree in terms of ancestor nodes without explicitly modeling each node individually. A real-world multi-product, multi-period network planning optimization model is used to illustrate the effectiveness of the proposed implementation approach.

Keywords: Multi-Period Optimization, Stochastic Programming, Extensive Form Generation.

1 Introduction

Stochastic programming (SP) with recourse is a powerful framework for modeling mathematical optimization problems that contain uncertain parameters. In this framework, it is assumed that the random vector has a joint probability distribution. In order to solve practical problems, the joint probability distribution governing the uncertain data is *discretized*. This results in scenario trees, which are the main inputs to SP models. Standard references on the theory and applications of SP include Prékopa (1995); Birge & Louveaux (2011); King

*Center for Advanced Process Decision-making (CAPD). Department of Chemical Engineering. Carnegie Mellon University. Pittsburgh, PA, 15213, USA.

& Wallace (2012). In this paper, we focus on SP with exogenous uncertainty, i.e., when the decisions do not influence the realization of the uncertain parameters.

There are two major classes of SP models: two- and multi-stage. In the former, the first-stage, or *here-and-now*, decisions are taken before the uncertainty is revealed, while the second-stage, or *wait-and-see*, decisions are taken after the uncertain parameters assume their values. In multi-stage problems, the uncertainty is revealed in multiple stages, and decisions are taken at every stage. Moreover, the decisions must satisfy the *non-anticipativity condition* (NAC), which means that decisions must be the same over indistinguishable scenarios at a given stage. In its simplest form, the SP model is solved through its deterministic equivalent form (also called *extensive form*), which can be a mixed-integer nonlinear programming model in the most general case.

The extensive form of the SP model can be generated *implicitly* or *explicitly* with respect to the NAC. In the explicit model, variables for all scenarios and stages are generated, and the non-anticipativity constraints are explicitly added to the model. In contrast, the implicit model generates only the variables and constraints for scenarios that are distinguishable at a given stage. The implicit model leads to considerable reduced size and memory requirements in large-scale applications. While presolving techniques of MILP codes can successfully reduce the problem size to its implicit extensive form, generating the explicit model may exceed memory requirements due to the large number of non-anticipativity conditions. This difficulty is avoided with the implicit extensive form that is proposed in this paper.

This paper is organized as follows. In Section 2, we review relevant previous work on implicit extensive form generation of stochastic programs. Section 3 defines the problem addressed in this paper, and introduces a motivating example to illustrate the generation of stochastic programs. In Section 4, we lay out some preliminaries and notation with respect to multi-period optimization models and scenario trees. Section 5 describes and illustrates the auxiliary parameters used to obtain a reduced-size implicit extensive form of the stochastic programming model. It also presents the general optimization formulation for the resulting implicit extensive form, and it contains an analysis to quantify the difference in problem size for the explicit and the implicit formulations. Section 6 presents a motivating production planning example. Section 7 presents the results obtained with the proposed approach for a large-scale industrial example. A summary is presented in Section 8.

2 Previous Work

Birge (1985) recognized that the Deterministic Equivalent Problem (DEP) of Stochastic Programming (SP) models is usually very large, and standard solution procedures are costly. The author described an outer-linearization-based decomposition algorithm called Nested Benders' Decomposition (NBD) that uses a dynamic programming representation of the multi-stage stochastic linear program with fixed recourse. This representation uses the structure of the scenario tree to only generate variables and constraints corresponding to each node at a given time period or stage. Consequently, the non-anticipativity condition (NAC) is implicitly taken into account. The efficiency of the NBD algorithm has been evaluated in several works (Gassmann, 1990; Dempster & Thompson, 1998; Parpas & Rustem, 2007). AIMMS (Roelofs & Bisschop, 2013) offers an implementation of the NBD algorithm.

To the best of our knowledge, the first work on proposing a modeling approach (i.e., at the mathematical formulation level) for the generation of the implicit extensive form of SP models is by Gassmann & Ireland (1995). The authors describe the use of parameters and index sets to formulate finite event trees in Algebraic Modeling Languages (AMLs). More specifically, the authors consider a scenario tree as consisting of a *base scenario* and one or more additional scenarios. A base scenario has all state variables defined for all time periods, including the root (present) node. Each additional scenario shares at least the root period problem state with the base scenario and branches from a *parent* scenario so that it has a distinct new problem state at its *start time*, some time period after the initial period. These tree events are captured by parameters and index sets that are properly initialized.

Following their previous work, the same authors proposed additional extensions to AMLs that cover chance-constrained programming as well as recourse problems (Gassmann & Ireland, 1996). The latter includes scenario- and distribution-based problems. The authors show specific examples in AMPL (Fourer, Gay, & Kernighan, 2013).

The approach described in this paper shares some similarities with the one proposed by Gassmann & Ireland (1995). In our approach, the redundant variables and constraints are suppressed from the final SP model through the use of auxiliary parameters that capture the structure of the scenario tree. Before describing the approach, we define the structure of the multi-period DEP problem in the next section.

3 Problem Definition

The objective of this paper is to describe an approach for implementing stochastic programs at the level of mathematical formulations. This not only gives the modeler full access to the generated stochastic model, but also makes it platform-independent.

The premises and outcomes of the proposed implementation approach are as follows:

- Given:
  - Deterministic multi-period optimization model;
  - Scenario tree information that represents the uncertainty.
- Generate:
  - Implicit extensive form of the stochastic program.

A scenario tree is a graphical representation of the different realizations of the uncertain parameters, and their associated probabilities. A sequence of realizations and their probabilities from the beginning to the end of the time horizon under consideration is called a scenario. Given the numerical values of the uncertain parameters and the time (stage) at which they occur, one can create a binary parameter (or an index set) that represents the non-anticipativity condition, i.e., in which stage the decisions of two scenarios must be the same. This is all that is needed to generate the explicit extensive form of the SP model. In order to generate the implicit extensive form, we propose two other sets of parameters to restrict the generation of variables and constraints only for indistinguishable scenarios at a given stage.

4 Preliminaries and Notation

This section establishes the nomenclature and notation with regards to deterministic and stochastic programming (SP) multi-period optimization models, and scenario trees.

4.1 Multi-Period Optimization Models

The focus of this paper is on optimization models that span multiple time periods. When uncertain parameters are considered, these models can either be formulated as two- or multi-stage stochastic programs. When modeled as two-stage stochastic programs, multiple periods are lumped into the same stage. This can be viewed as an approximation of the multi-stage model in which each stage corresponds to a time period. Figure 1 illustrates the two- and multi-stage scenario trees, which are used to represent uncertainty in SP models.

A general *explicit* extensive model is given in (1). The notation is as follows. Let $t \in T$ denote time periods and $j \in J$ represent scenarios. The decision variables include $z_{t,j}$ and w, where both can be vectors of variables, and may have additional indices. The main distinction between these variables is that z is defined over time periods, and thus may assume distinct values for different scenarios j at time period t, whereas w is not indexed by t and assumes the same value for all periods and scenarios. Let $\xi_{t,j}$ be uncertain parameters, and p_j the probability of scenario j. Additional indices are not shown in order to keep the presentation clean, and emphasize the different modeling aspects for terms that are dependent and independent of t. However, additional indices, such as products, plants, regions, customers, etc., directly contribute to the total problem size.

$$\min_{z_{t,j},\, w} \ \beta(w) + \sum_{j \in J} \mathrm{p}_j \sum_{t \in T} \alpha_{t,j}(z_{t,j}, w; \xi_{t,j}) \tag{1a}$$

$$\text{s.t.} \quad \gamma_{t,j}(z_{t,j}, w; \xi_{t,j}) + \psi_{t-\tau,j}(z_{t-\tau,j}, w; \xi_{t-\tau,j}) \leq 0 \qquad \forall\, t \in T,\ j \in J \tag{1b}$$

$$\phi(w) + \sum_{t \in T} \sum_{\substack{j' \in J \\ j' = j}} \delta_{t,j'}(z_{t,j'}, w; \xi_{t,j'}) \leq 0 \qquad \forall\, j \in J \tag{1c}$$

$$z_{t,j} = z_{t,j'} \qquad \forall\, t \in T,\ (j, j') \in J,\ j' = j + 1,\ \mathrm{NAC}_{t,j,j'} = 1 \tag{1d}$$

$$z_{t,j} \in Z \qquad \forall\, t \in T,\ j \in J \tag{1e}$$

$$w \in W \tag{1f}$$

where $\alpha_{t,j}(\cdot,\cdot;\cdot)$, $\gamma_{t,j}(\cdot,\cdot;\cdot)$, $\psi_{t-\tau,j}(\cdot,\cdot;\cdot)$, and $\delta_{t,j}(\cdot,\cdot;\cdot)$ are time-indexed continuous functions, $\beta(\cdot)$ and $\phi(\cdot)$ are continuous functions that do not depend on the time periods, and Z and W denote sets of continuous (real) and/or discrete (integer, binary) variables. Therefore, we separate functional terms in the objective function and constraints that depend on the time periods from those that are not indexed by t. In constraints (1b), τ is any positive integer between 1 and $|T| - 1$.

We note that it is usually the case in multi-period optimization models that $\tau = 1$; thus, constraints (1b) link decisions made at time period t and its immediate previous time period $t - 1$. An example of such constraints are inventory balances that relate the inventory levels

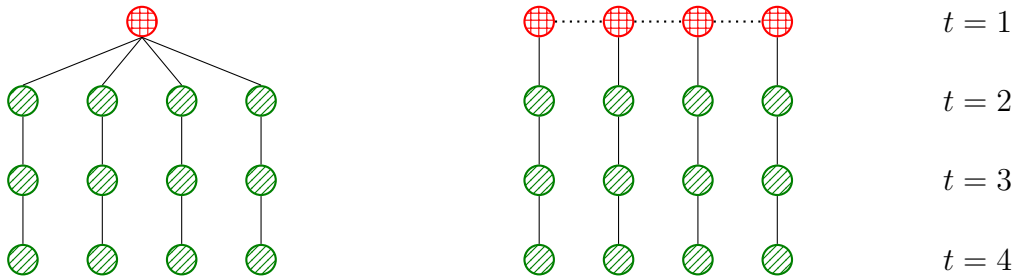
across adjacent time periods. However, the notation and proposed implicit extensive form are kept general for any previous time period $t - \tau$.

The binary parameter $\text{NAC}_{t,j,j'}$ in constraints (1d) enforces the non-anticipativity condition. It takes the value of 1 if decisions at stage t must be the same for scenarios j and j', or 0 otherwise. It can be initialized from the data contained in the scenario tree and its associated sequence of decisions, which are discussed in the next section.

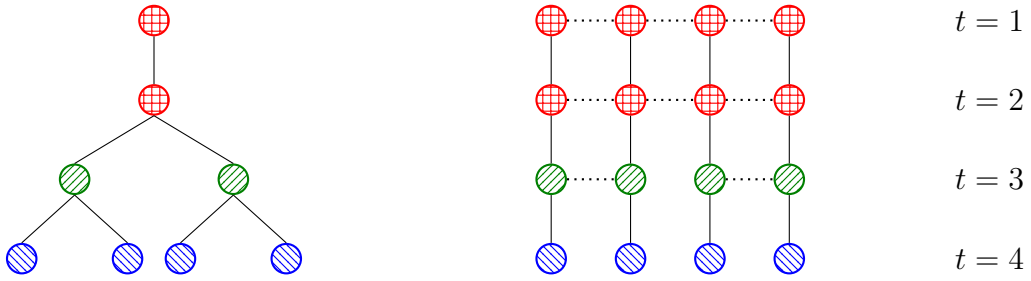
4.2 Scenario Trees

The structure of a stochastic program is dictated by the information contained in the scenario tree, which is fixed and known *a priori* when only exogenous uncertainty is considered. A scenario tree is a discrete representation of the joint probability distribution of the random vector. It consists of nodes and branches or outcomes. Each node represents the state of the problem at a particular instant when decisions are made, whereas the branches represent the different realizations of the random variables. Nodes at the same level belong to the same time period t.

Figure 1 shows multi-period two- and multi-stage scenario trees and their corresponding sequences of decisions (Ruszczyński, 1997). Note that both trees in Figures 1(a) and 1(b) represent stochastic processes for 4 time periods, and have 4 scenarios. A *scenario* is a path from the root node to a leaf node.



(a) Two-stage scenario tree (left) and its sequence of decisions (right). Nodes are colored according to the classical definition of stages: stage 1, stage 2.



(b) Multi-stage scenario tree (left) and its sequence of decisions (right). Nodes are colored according to the classical definition of stages: stage 1, stage 2, and stage 3.

Figure 1: Multi-period scenario trees and their sequences of decisions. Dotted lines represent the non-anticipativity condition.

To simplify the notation in the implementation approach described in the next section, from this point onward *a stage corresponds to a time period* t. Therefore, the trees in Figure 1 can be interpreted as follows:

- The tree in Figure 1(a) has *4 stages* where the decisions in stage 1 must be the same for every scenario (NAC), but the decisions in stages $t = \{2,\ 3,\ 4\}$ do not have to be the same among different scenarios.
- The tree in Figure 1(b) has *4 stages* where the NAC is as follows: the decisions in stages 1 and 2 must be the same for every scenario, the decisions in stage 3 for scenarios 1 and 2 have to be the same, the decisions in stage 3 for scenarios 3 and 4 have to be the same, but the decisions in stages $t = \{3,\ 4\}$ do not have to be the same among different scenarios.

5 Proposed Generation Approach

In this section, we introduce the three sets of parameters that act in conjunction to generate a memory-efficient SP model. Then, we present the implicit extensive form of the general stochastic model defined in (1). We note that the proposed approach operates at the level of the mathematical formulation of the optimization model, i.e., it precedes the stage of translating the algebraic model into a file structure that a specific optimization solver can process.

5.1 Auxiliary Parameters

The first parameter, $\text{NAC}_{t,j,j'}$, is introduced for defining the explicit extensive form in equation (1), and is responsible for enforcing the non-anticipativity condition (NAC). The tree in Figure 1(b) is used to numerically illustrate its definition and the result is shown in Table 1. Note that there are other possible ways of initializing the parameter $\text{NAC}_{t,j,j'}$. For instance, for a given stage t, if scenario $j = 1$ is indistinguishable from scenario $j' = 2$, which in turn is indistinguishable from scenario $j'' = 3$, then we do not have to explicitly state that scenarios $j = 1$ and $j'' = 3$ are indistinguishable by setting $\text{NAC}_{t,1,3} = 1$. However, the initialization scheme used in the proposed approach is beneficial when considering the two other sets of parameters in conjunction to obtain the desired generated stochastic model.

j	j'			
	1	2	3	4
1	1	1	1	1
2	·	1	1	1
3	·	·	1	1
4	·	·	·	1

(a) $\text{NAC}_{1,j,j'}$

j	j'			
	1	2	3	4
1	1	1	1	1
2	·	1	1	1
3	·	·	1	1
4	·	·	·	1

(b) $\text{NAC}_{2,j,j'}$

j	j'			
	1	2	3	4
1	1	1	·	·
2	·	1	·	·
3	·	·	1	1
4	·	·	·	1

(c) $\text{NAC}_{3,j,j'}$

j	j'			
	1	2	3	4
1	1	·	·	·
2	·	1	·	·
3	·	·	1	·
4	·	·	·	1

(d) $\text{NAC}_{4,j,j'}$

Table 1: Initialization of $\text{NAC}_{t,j,j'}$ that models the non-anticipativity condition implied by the scenario tree in Figure 1(b). The dots represent zeros.

The second parameter, $\text{Sc2St}_{t,j}$, is a binary parameter that accounts for the mapping of scenarios to stages, i.e., it takes the value of 1 if scenario j is "unique" in stage t, or 0 otherwise. This parameter ensures that only variables and constraints for distinguishable scenarios at a given stage are generated, thus avoiding creating variables and constraints for *all* scenarios at every stage, and then equating variables according to the NAC. The efficient reduction in problem size is due to this parameter. Again, the tree in Figure 1(b) is used to illustrate the initialization of this parameter and the result is shown in Table 2. Note that, for instance, for $t = \{1,\ 2\}$ only variables and constraints for scenario 1 may be generated.

t	j			
	1	2	3	4
1	1	·	·	·
2	1	·	·	·
3	1	·	1	·
4	1	1	1	1

Table 2: Entries of $\text{Sc2St}_{t,j}$ that correspond to the scenario tree in Figure 1(b). The dots represent zeros.

Finally, the third parameter, $\text{PaSc}_{t,j}$, is an *element* parameter whose range is the set of scenarios, and contains information about the parent of a given scenario in a stage. That is, given a scenario j and stage t, it returns the parent scenario that is unique as per the information contained in parameter $\text{Sc2St}_{t,j}$. Identifying the parent scenario that is unique is needed in cases where variables indexed by $t - \tau$ appear in constraints defined for each t, where τ is a positive integer. For instance, the nodes of scenarios $j = 1$ and $j' = 2$ and at stage $t = 4$ share the same parent node at stage $t' = 3$. Thus, only one scenario can be used to represent this parent node. By convention, we assign the smallest scenario index of the children nodes ($j = 1$) to the parent node. Therefore, $\text{PaSc}_{4,1} = \text{PaSc}_{4,2} = 1$. Similarly, nodes of scenarios $j = 3$ and $j' = 4$ at stage $t = 4$ share the same parent at stage $t' = 3$, and we set $\text{PaSc}_{4,3} = \text{PaSc}_{4,4} = 3$. Table 3 shows the initialized entries of this third parameter based on the tree in Figure 1(b).

t	j			
	1	2	3	4
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	3	3

Table 3: Entries of $\text{PaSc}_{t,j}$ that correspond to the scenario tree in Figure 1(b). The parent scenario of nodes in $t = 1$ is scenario 1 by convention.

Note that the three sets of parameters defined in this section are used together in the generation of the extensive form of the SP model, and their initialization follows a hierarchy: $\text{NAC}_{t,j,j'}$ is initialized first, then it is used to initialize $\text{Sc2St}_{t,j}$, which in turn has the necessary information to initialize $\text{PaSc}_{t,j}$.

5.2 Memory-Efficient Implicit Extensive Form

After properly initializing the three sets of parameters described in the previous section, the proposed memory-efficient extensive form can be written as in equation (2). The expressions involving the three sets of parameters are colored in red.

$$\min_{z_{t,j},\, w} \beta(w) + \sum_{j \in J} \mathrm{p}_j \sum_{t \in T} \sum_{\substack{j' \in J \\ \mathrm{NAC}_{t,j',j}=1 \\ \mathrm{Sc2St}_{t,j'}=1}} \alpha_{t,j'}(z_{t,j'}, w; \xi_{t,j'}) \tag{2a}$$

s.t.

$$\gamma_{t,j}(z_{t,j}, w; \xi_{t,j}) + \psi_{t-\tau,j'}(z_{t-\tau,j'}, w; \xi_{t-\tau,j'}) \leq 0 \qquad \forall\, t \in T,\ j \in J,\ \mathrm{Sc2St}_{t,j} = 1,\ j' = \mathrm{PaSc}_{t-\tau+1,j} \tag{2b}$$

$$\phi(w) + \sum_{t \in T} \sum_{\substack{j' \in J \\ \mathrm{NAC}_{t,j',j}=1 \\ \mathrm{Sc2St}_{t,j'}=1}} \delta_{t,j'}(z_{t,j'}, w; \xi_{t,j'}) \leq 0 \qquad \forall\, j \in J \tag{2c}$$

$$z_{t,j} \in Z \qquad \forall\, t \in T,\ j \in J,\ \mathrm{Sc2St}_{t,j} = 1 \tag{2d}$$

$$w \in W \tag{2e}$$

Note that the implicit memory-efficient model has essentially the same structure as of the explicit extensive form in equation (1). The explicit model has the potential to become very large for real-world applications since it includes (redundant) variables and constraints for all scenarios in every stage. The main purpose of the implicit formulation is to exploit the structure of the scenario tree to only generate variables and constraints for distinguishable scenarios at a given stage. The explicit extensive form is the benchmark used to evaluate the performance of proposed implementation approach to generate an implicit extensive form.

The resulting implicit extensive form model has a reduced size due to the three sets of auxiliary parameters described previously, and the absence of the explicit non-anticipativity constraints. This highlights the main advantage of the proposed approach since it acts at the level of the mathematical formulation, which can be implemented in any optimization modeling platform. The procedures in pseudo-code for initializing the three sets of parameters that are used in the proposed approach are presented in Appendix A. In the next subsection, we present an analysis of the reduction in model size that is achieved with the proposed implementation approach.

5.3 Model Size Reduction Analysis

Intuitively, the implicit extensive form of the stochastic program yields models with fewer variables and constraints than the explicit formulation due to the binary auxiliary parameters that restrict the generation of redundant variables and constraints. In this section, we show expressions that relate the number of variables and constraints in both formulations.

We begin the analysis with respect to the number of variables. Both implicit and explicit models share the same number of variables that are not indexed by time, if any. The difference is then due to the number of time-indexed variables. In the implicit model, only variables that represent indistinguishable scenarios at a given stage are generated, which is achieved by using the binary auxiliary parameter $\text{Sc2St}_{t,j}$ in time-indexed constraints, such as constraints (2b).

The time-indexed constraints in the explicit model (see constraints (1b)) contain $k \cdot |T| \cdot |J|$ time-indexed variables, where $|T|$ ($|J|$) denotes the cardinality of set T (J), and k is a constant that represents the summation of the cardinality of sets other than T and J, if any, of all time-indexed variables. For instance, suppose the stochastic optimization model has two sets of variables, $x_{m,t,j}$ and $y_{n,t,j}$. Therefore, $k = |M| + |N|$, which yields $(|M| + |N|) \cdot |T| \cdot |J|$ time-indexed variables. In the implicit model, only time-indexed variables whose t-th and j-th entries for which $\text{Sc2St}_{t,j} = 1$ are generated. In other words, only the elements of the set $T \times J$ for which $\text{Sc2St}_{t,j} = 1$ are used to generate variables. Therefore, the number of time-indexed variables in the implicit model is $k \cdot |\text{Sc2St}_{t,j}|$, where $|\text{Sc2St}_{t,j}|$ denotes the summation of all nonzero entries of the parameter $\text{Sc2St}_{t,j}$. Note that the constant k is the same for both models, and it vanishes when taking the ratio of the number of variables in both models. The ratio between the number of variables in both models can be written as follows:

$$\frac{(\#\text{ time-indexed variables Explicit})}{(\#\text{ time-indexed variables Implicit})} = \frac{|T| \cdot |J|}{|\text{Sc2St}_{t,j}|} \tag{3}$$

Since $|T| \cdot |J| > |\text{Sc2St}_{t,j}|$, the explicit model has more variables than the implicit model.

Similarly for the number of variables, both explicit and implicit models share the same number of constraints that are not indexed by time. Therefore, the analysis for the difference in number of constraints can be restricted to time-indexed constraints. A clear distinction between the two models is the absence of the non-anticipativity constraints in the implicit model. The binary auxiliary parameter $\text{NAC}_{t,j,j'}$ captures the non-anticipativity condition, which depends on the structure of the scenario tree. Let us define the total number of pairs of indistinguishable scenarios for all stages as follows:

$$ScenPairs = \sum_{t \in T} \sum_{j \in J} \sum_{\substack{j' \in J,\ j'=j+1 \\ \text{NAC}_{t,j,j'}=1}} (1) \tag{4}$$

In other words, $ScenPairs$ represents the number of constraints in the explicit definition of the non-anticipativity condition (see constraints (1d)).

Similar to the number of variables, the total number of time-indexed constraints in the explicit model (see constraints (1b)) is given by $k_1 \cdot |T| \cdot |J|$, where k_1 is a constant that represents the summation of the cardinality of sets other than T and J, if any, of all time-indexed constraints. For the implicit model, the number of time-indexed constraints is given by $k_1 \cdot |\text{Sc2St}_{t,j}|$. The relationship between the number of constraints in both models can be written as follows:

$$\begin{aligned}(\#\text{ constraints Explicit}) - (\#\text{ constraints Implicit}) = \\ k_1 \cdot (|T| \cdot |J| - |\text{Sc2St}_{t,j}|) + k_2 \cdot ScenPairs\end{aligned} \tag{5}$$

where k_2 is a constant that represents the summation of the cardinality of sets other than T and J, if any, of all non-anticipativity constraints. Note that the left-hand side accounts for the total number of constraints in each model. Since the number of constraints that are not indexed by time is the same in both models, it cancels out in the subtraction. Again, $|T| \cdot |J| > |\mathrm{Sc2St}_{t,j}|$, which implies that there is also a reduction in the number of constraints. In practice, the last term on the right-hand side may account for half or more of the total number of constraints in the explicit model.

In the next section, we present a motivating production planing example to illustrate the proposed approach to generate the implicit extensive form.

6 Motivating Production Planning Example

The *explicit* extensive form of the stochastic programming model is given in equation (6). For illustration purposes, we only consider the demand to be uncertain, thus it is the only parameter that is indexed by j (scenarios).

$$\min \ \mathrm{v}y + \sum_{j \in J} \mathrm{p}_j \sum_{t \in T} (\mathrm{c}_t x_{t,j} + \mathrm{h}_t s_{t,j}) \tag{6a}$$

s.t.

$$s_{t,j} = s_{t-1,j} + x_{t,j} - \mathrm{d}_{t,j} \qquad \forall\, t \in T,\ j \in J \tag{6b}$$

$$y \geq \mathrm{y}^L - \sum_{t \in T} x_{t,j} \qquad \forall\, j \in J \tag{6c}$$

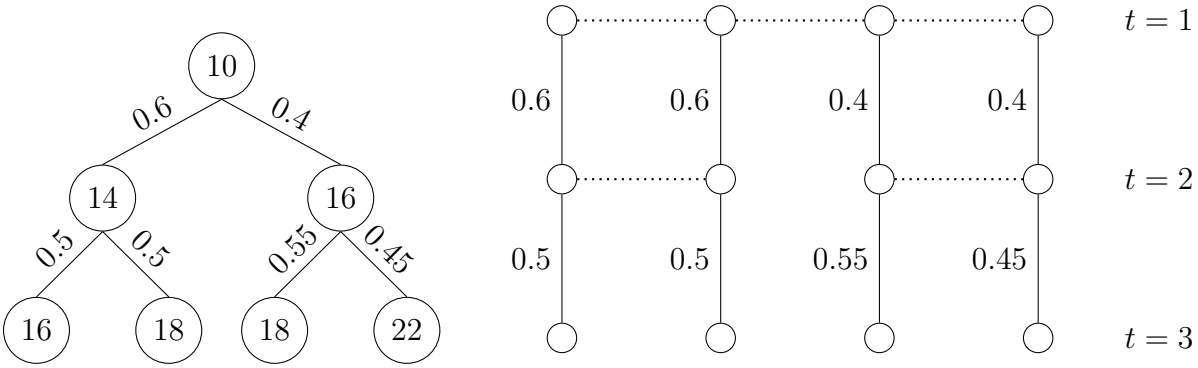
$$x_{t,j},\ s_{t,j},\ y \geq 0 \qquad \forall\, t \in T,\ j \in J \tag{6d}$$

$$x_{t,j} = x_{t,j'} \qquad \forall\, t \in T,\ (j, j') \in J,\ j' = j + 1,\ \mathrm{NAC}_{t,j,j'} = 1 \tag{6e}$$

$$s_{t,j} = s_{t,j'} \qquad \forall\, t \in T,\ (j, j') \in J,\ j' = j + 1,\ \mathrm{NAC}_{t,j,j'} = 1 \tag{6f}$$

where the objective function (6a) determines the total expected cost to be minimized (c_t, h_t, and v are unit production, storage, and contract violation penalty costs, respectively, x_t, s_t, and y are production, storage, and contract amounts, respectively), constraints (6b) represent the inventory balance ($\mathrm{d}_{t,j}$ represents product demand), constraints (6c) set a lower bound on the contract amounts (y^L represents minimum contract amount), constraints (6d) contain the non-negativity property (bounds) of the decision variables, and constraints (6e) and (6f) are the non-anticipativity constraints. Data for the parameters are available in Appendix B.

The three-stage scenario tree with node values (demand) and probabilities (values over arcs), and its corresponding sequence of decisions are given in Figure 2.



(a) Scenario tree.

(b) Sequence of decisions.

Figure 2: Scenario tree (node values represent demand realizations) and corresponding sequence of decisions for the motivating example.

The data values for demand in each stage and scenario, $d_{t,j}$, correspond to the structure of the scenario tree and its associated sequence of decisions. Those values are used to initialize the three sets of auxiliary parameters of the proposed implicit formulation. The auxiliary parameters are initialized as shown in Tables 4 to 6.

j	j'			
	1	2	3	4
1	1	1	1	1
2	·	1	1	1
3	·	·	1	1
4	·	·	·	1

(a) $NAC_{1,j,j'}$

j	j'			
	1	2	3	4
1	1	1	·	·
2	·	1	·	·
3	·	·	1	1
4	·	·	·	1

(b) $NAC_{2,j,j'}$

j	j'			
	1	2	3	4
1	1	·	·	·
2	·	1	·	·
3	·	·	1	·
4	·	·	·	1

(c) $NAC_{3,j,j'}$

Table 4: Initialization of $NAC_{t,j,j'}$ that models the non-anticipativity condition implied by scenario tree in Figure 2(a). The dots represent zeros.

t	j			
	1	2	3	4
1	1	·	·	·
2	1	·	1	·
3	1	1	1	1

Table 5: Entries of $Sc2St_{t,j}$ that correspond to the scenario tree in Figure 2(a). The dots represent zeros.

t	j			
	1	2	3	4
1	1	1	1	1
2	1	1	1	1
3	1	1	3	3

Table 6: Entries of $\mathrm{PaSc}_{t,j}$ that correspond to the scenario tree in Figure 2(a). The parent scenario of nodes in $t = 1$ is scenario 1 by convention.

The memory-efficient, *implicit* extensive form of the stochastic programming model is given in equation (7). The expressions involving the three sets of parameters are colored in red.

$$\min \ \mathrm{v}y + \sum_{j\in J} \mathrm{p}_j \sum_{t\in T} \sum_{\substack{j'\in J \\ \mathrm{NAC}_{t,j',j}=1 \\ \mathrm{Sc2St}_{t,j'}=1}} (\mathrm{c}_t x_{t,j'} + \mathrm{h}_t s_{t,j'}) \tag{7a}$$

s.t.

$$s_{t,j} = s_{t-1,j'} + x_{t,j} - \mathrm{d}_{t,j} \qquad \forall\, t \in T,\ j \in J,\ \mathrm{Sc2St}_{t,j} = 1,\ j' = \mathrm{PaSc}_{t,j} \tag{7b}$$

$$y \geq \mathrm{y}^L - \sum_{t\in T} \sum_{\substack{j'\in J \\ \mathrm{NAC}_{t,j',j}=1 \\ \mathrm{Sc2St}_{t,j'}=1}} x_{t,j'} \qquad \forall\, j \in J \tag{7c}$$

$$x_{t,j},\ s_{t,j},\ y \geq 0 \qquad \forall\, t \in T,\ j \in J,\ \mathrm{Sc2St}_{t,j} = 1 \tag{7d}$$

All models were implemented in AIMMS 3.13, and solved with GUROBI 5.1 on a desktop computer with the following specifications: Dell Optiplex 990 with 4 Intel® Core™ i7-2600 CPUs at 3.40 GHz (total 8 threads), 8 GB of RAM, and running Windows 7 Enterprise.

The implicit formulation has fewer variables and constraints than the explicit model as shown in Table 7. The numbers obtained with the approach proposed in this paper are under the column "Implicit", whereas column "Explicit" corresponds to the results obtained with the extensive form model by explicitly adding the non-anticipativity constraints. In particular, the non-anticipativity constraints account for half of the total number of constraints in the explicit extensive model. The effect in memory usage was not significant in this small example. All problems were solved in less than one second. It is interesting to note that, in this example, GUROBI's presolve reduced the "Explicit" model to the same size as the "Implicit" model.

Table 7: Motivating example and model sizes.

	Deterministic	**Stochastic**	
		Implicit (This Work)	Explicit
Constraints	4	11	26
Variables	7	15	25

We now use the relationships developed in Subsection 5.3 to confirm the results above. There is one variable, y, that is not indexed by time in both models. The number of nonzero entries in $\text{Sc2St}_{t,j}$ is 7 (see Table 5). Also, $|T|\cdot|J| = 3\cdot 4 = 12$. Thus, the ratio of time-indexed variables in both models can be calculated using equation (3) as follows:

$$\frac{(\#\text{ time-indexed variables Explicit})}{(\#\text{ time-indexed variables Implicit})} = \frac{|T|\cdot|J|}{|\text{Sc2St}_{t,j}|} = \frac{12}{7}$$

From Table 7, there are 24 and 14 time-indexed variables in the explicit and implicit models, respectively. Their ratio can be simplified to $\frac{12}{7}$.

The only time-indexed constraints are (6b), and since they contain no other indices than t (time) and j (scenario), the constant k_1 in equation (5) is equal to 1. In addition, because there are two time-indexed variables, $x_{t,j}$ and $s_{t,j}$, there are two sets of non-anticipativity constraints (constraints (6d) and (6e)). Since those variables are only indexed by t and j, the value of k_2 is 2, which accounts for the two sets of non-anticipativity constraints. Also, the number of pairs of indistinguishable scenarios, $ScenPairs$ (see equation (4)) is 5. Thus, the difference in total number of constraints of both models can be calculated using equation (5) as follows:

$$\begin{aligned}(\#\text{ constraints Explicit}) - (\#\text{ constraints Implicit}) \\ &= k_1 \cdot (|T|\cdot|J| - |\text{Sc2St}_{t,j}|) + k_2 \cdot ScenPairs \\ &= (1)\cdot(12-7) + (2)\cdot(5) = 15\end{aligned}$$

From Table 7, there are 26 and 11 constraints in the explicit and implicit models, respectively. Their difference is 15.

In the next section, we show the numerical results of a large-scale industrial test case to illustrate the memory-efficiency advantage of the proposed implementation approach.

7 Industrial Test Case

The industrial test case concerns the optimal production planning of a network of production sites. Each site contains several plants that are highly integrated. The plants can also transfer products between sites. At these multiple production sites, with more than 25 production facilities, several products are manufactured. The time horizon of one year is divided into monthly time periods. The objective of the optimization model is to maximize profit.

The deterministic model contains only linear constraints and continuous variables. For the stochastic model, we consider the demand of a given product to be the uncertain parameter. By lumping some of the time periods into the same stage, the scenario tree for the stochastic model has 15 scenarios.

Table 8 shows the statistics of the optimization problems. The "Total Time" accounts for the loading time of the model into memory by the solver and its solving time. It is clear that it is advantageous to invest in a more careful implicit extensive form implementation for large-scale problems, which sometimes can be solved in a few seconds or minutes without the need to employ decomposition strategies. The explicit deterministic equivalent model of the multi-stage stochastic program could not be completely loaded by GUROBI as it exceeded

the memory resources, and therefore the presolve could not be applied to reduce the model size. This problem is avoided with the implicit formulation. We note that the majority of the constraints in the explicit extensive form is due to the NAC.

Table 8: Industrial test case and model sizes.

	Deterministic	**Stochastic**	
		Implicit (This Work)	Explicit
Constraints	207,720	1,143,484	19,730,911
Variables	304,397	1,674,161	4,591,445
Solving Time [s]	1.00	12.85	*
Total Time [s]	6.96	41.76	*
Memory Usage [GB]	0.35	2.11	> 8.00

* Not enough memory to extract the model to the solver.

8 Conclusions

In this paper, we described a memory-efficient approach for the formulation and implementation of stochastic programming models. The main advantage of the proposed approach is that it allows the generation of deterministic equivalent or extensive form of stochastic programming models of reduced size by performing minor modifications at the level of the mathematical formulation. Therefore, it is modeling-platform and programming-language independent.

The key aspects of the approach are three sets of auxiliary parameters that ensure that only variables and constraints that represent distinguishable scenarios at a given stage are generated. In addition, the modeler/user only needs to provide initialized uncertain parameters, since the structure of the tree is captured via the proper initialization of the three sets of auxiliary parameters. Lastly, the approach accounts for the case of variables and constraints that are not indexed by time periods and variables that refer to previous time periods in some constraints.

Acknowledgments

The author gratefully acknowledges financial support from The Dow Chemical Company, as well as Dr. John Wassick and Dr. Anshul Agarwal for the discussions concerning the real-world planning model and the industrial test case data.

References

Birge, J. R., and Louveaux, F. 2011. *Introduction to Stochastic Programming*. Springer, second edition.

Birge, J. R. 1985. Decomposition and Partitioning Methods for Multistage Stochastic Linear Programs. *Operations Research*. 33(5):989–1007.

Dempster, M. A. H., and Thompson, R. T. 1998. Parallelization and Aggregation of Nested Benders Decomposition. *Annals of Operations Research.* 81(0):163–188.

Fourer, R.; Gay, D. M.; and Kernighan, B. W. 2013. *AMPL: A Modeling Language for Mathematical Programming.* http://www.ampl.com/.

Gassmann, H. I., and Ireland, A. M. 1995. Scenario Formulation in an Algebraic Modelling Language. *Annals of Operations Research.* 59(1):45–75.

Gassmann, H. I., and Ireland, A. M. 1996. On the Formulation of Stochastic Linear Programs using Algebraic Modelling Languages. *Annals of Operations Research.* 64(1):83–112.

Gassmann, H. I. 1990. MSLIP: A Computer Code for the Multistage Stochastic Linear Programming Problem. *Mathematical Programming.* 47(1-3):407–423.

King, A. J., and Wallace, S. W. 2012. *Modeling with Stochastic Programming.* Springer Series in Operations Research and Financial Engineering. Springer Science+Business Media New York.

Parpas, P., and Rustem, B. 2007. Computational Assessment of Nested Benders and Augmented Lagrangian Decomposition for Mean-Variance Multistage Stochastic Problems. *INFORMS Journal on Computing.* 19(2):239–247.

Prékopa, A. 1995. *Stochastic Programming.* Mathematics and Its Applications. Kluwer Academic Publishers.

Roelofs, M., and Bisschop, J. 2013. *Advanced Interactive Multidimensional Modeling System (AIMMS).* http://www.aimms.com/.

Ruszczyński, A. 1997. Decomposition Methods in Stochastic Programming. *Mathematical Programming* 79(1-3):333–353.

Appendix A Initialization of Auxiliary Parameters

There may be multiple uncertain parameters under consideration. Every element of the random vector does not have to assume different values for each outcome at each stage. We denote the set of all uncertain parameters under consideration by $\Xi_{t,j}$, for $t \in T$ and $j \in J$. The procedures described in this appendix initialize the three sets of auxiliary parameters that are used to generate the implicit extensive form of stochastic programs. The workflow is as follows: the modeler initializes each uncertain parameter, $\xi_{t,j} \in \Xi_{t,j}$, and then calls the procedures in the given order as explained below.

First, the binary parameter $\text{NAC}_{t,j,j'}$ is initialized in Algorithm A.1. Each uncertain parameter in the optimization model is used to identify distinguishable scenarios at a given

stage in the entire tree.

Input : Uncertain parameter(s), $\xi_{t,j} \in \Xi_{t,j}$
Output: Initialized $\text{NAC}_{t,j,j'}$

0. Empty output parameter by setting $\text{NAC}_{t,j,j'} := 0$
1. Begin initialization: for each t the pair of scenarios (j, j') is distinguishable if $j < j'$ $\text{NAC}_{t,j,j'} := 1$ where $j \leq j'$
2. Use uncertain parameters to detect distinguishable scenarios

```
for [t ∈ T, (j, j') ∈ J] such that (t > 1) and (j < j') do
    for [ξ_{t,j} ∈ Ξ_{t,j}] do
        AreScenariosDifferent := false
        for t' ∈ T such that t' ≤ t do
            if [ξ_{t',j} ≠ ξ_{t',j'}] or [ξ_{t',j} = ξ_{t',j'} and NAC_{t,j,j'} = 0] then
                // Scenarios j and j' are distinguishable
                NAC_{t,j,j'} := 0
                AreScenariosDifferent := true
                break
            end
        end
        if not AreScenariosDifferent then
            NAC_{t,j,j'} := 1
        end
    end
end
```

Algorithm A.1: Procedure `CreateNAC`: Pseudo-code for initializing the binary parameter $\text{NAC}_{t,j,j'}$.

Then, the binary parameter $\text{Sc2St}_{t,j}$ is initialized as shown in Algorithm A.2. The notation $T_{[n]}$ denotes the n-th element of ordered set T.

Input : Initialized $\text{NAC}_{t,j,j'}$
Output: Initialized $\text{Sc2St}_{t,j}$

0. Empty output parameter by setting $\text{Sc2St}_{t,j} := 0$
1. Only the first scenario is unique in the first stage
   $\text{Sc2St}_{T_{[1]},J_{[1]}} := 1$
2. Use previously initialized $\text{NAC}_{t,j,j'}$ to map unique scenarios to stages
   for $t \in T$ such that $t \geq T_{[2]}$ **do**
      `CurrScenario` $:= J_{[1]}$ `// Get first scenario`
      repeat "RepeatLoop"
         while `CurrScenario` $< J_{[|J|]}$ **do**
            for $j' \in J$ such that $j' >$ `CurrScenario` **do**
               `// Only include the distinguishable scenarios in each stage`
               if not $\text{NAC}_{t,\texttt{CurrScenario},j'}$ **then**
                  `CurrScenario` $:= j'$
                  $\text{Sc2St}_{t,j'} := 1$
                  break
               end
               break "RepeatLoop" **when** $(j' = J_{[|J|]})$
            end
         end
         break "RepeatLoop" **when** (`CurrScenario` $= J_{[|J|]}$)
      until
   end

Algorithm A.2: Procedure `CreateScenariosToStages`: Pseudo-code for initializing the binary parameter $\text{Sc2St}_{t,j}$.

Finally, Algorithm A.3 shows the initialization of the element parameter $\text{PaSc}_{t,j}$.

Input : Initialized $\text{Sc2St}_{t,j}$
Output: Initialized $\text{PaSc}_{t,j}$

0. Empty output parameter by setting $\text{PaSc}_{t,j} := 0$
1. Parent scenario of first-stage scenario is the first scenario by convention
   $\text{PaSc}_{t,j} := J_{[1]}$ where $t = T_{[1]}$
   $\text{PaSc}_{t,j} := j$ where $t > T_{[1]}$ `// Will be overridden if applicable`
2. Parent scenario of second-stage scenario is the first scenario
   $\text{PaSc}_{t,j} := J_{[1]}$ where $t = T_{[2]}$
3. Set parent scenarios for remaining stages
   for $t \in T$ such that $t \geq T_{[3]}$ **do**
      `CurrStageScenario` $:= J_{[1]}$
      `PrevStageScenario` := `CurrStageScenario`
      $\text{PaSc}_{t,\texttt{CurrStageScenario}}$:= `PrevStageScenario`
      repeat "RepeatLoop"
         while `PrevStageScenario` $< J_{[|J|]}$ **do**
            for $j' \in J$ such that $j' >$ `PrevStageScenario` **do**
               `// Only include the distinguishable scenarios in each stage`
               if not $\text{Sc2St}_{t-1,j'}$ **then**
                  `CurrStageScenario` $:= j'$
                  $\text{PaSc}_{t,\texttt{CurrStageScenario}}$:= `PrevStageScenario`
               else
                  `PrevStageScenario` $:= j'$
                  break
               end
               break "RepeatLoop" **when** (`CurrStageScenario` $= J_{[|J|]}$)
            end
         end
         break "RepeatLoop" **when** (`PrevStageScenario` $= J_{[|J|]}$)
      until
   end

Algorithm A.3: Procedure `CreateParentScenario`: Pseudo-code for initializing the binary parameter $\text{PaSc}_{t,j}$.

The complete script for the memory-efficient implementation approach is summarized in Algorithm A.4, where the procedure `SolveStochasticModel` creates the mathematical program and calls the solver.

```
// Initialize auxiliary parameters
CreateNAC
CreateScenariosToStages
CreateParentScenario

// Generate and solve stochastic programming model
SolveStochasticModel
```

Algorithm A.4: Main script for generating the memory-efficient extensive form.

Appendix B Data for Motivating Example

The parameters for the production planning LP model are given in the following tables.

Table B.1: Unit production cost (c_t) [\$/t].

Time Period	**Group**
1	5
2	4
3	6

Table B.2: Unit storage cost (h_t) [\$/t].

Time Period	**Group**
1	1
2	1.5
3	1

Table B.3: Product demand ($d_{t,j}$) [t].

Time Period	**Scenario**			
	1	2	3	4
1	10	10	10	10
2	14	14	16	16
3	16	18	18	22

The contract violation penalty cost is $v = 2\,\$/t$, and the minimum contract amount is $y^L = 50\,t$.