Solving Stochastic Programs with Complete Integer Recourse: a framework using Gröbner Bases

Rüdiger Schultz
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Heilbronner Str. 10, D-10711 Berlin, Germany

Leen Stougie
Institute for Actuarial Sciences and Econometrics, University of Amsterdam
Roetersstraat 11, NL-1018 WB Amsterdam, The Netherlands

Maarten H. van der Vlerk
CORE, Université Catholique de Louvain
34 Voie du Roman Pays, B-1348 Louvain-la-Neuve, Belgium

Abstract

In this paper we present a framework for solving stochastic programs with complete integer recourse and discretely distributed right-hand side vector, using Gröbner basis methods from computational algebra to solve the numerous second-stage integer programs. Using structural properties of the integer expected recourse function, we prove that under mild conditions an optimal solution is contained in a finite set. Furthermore, we present a basic scheme to enumerate this set and suggest possible improvements to economize on the number of function evaluations needed.

1 Introduction

In this paper we are concerned with two-stage stochastic integer programs of the type

$$\min\{cx + Q(x) \ : \ x \in C\} \tag{1}$$

where

$$Q(x) = E_\xi v(Tx - \xi) \tag{2}$$

and

$$v(s) = \min\{\tilde{q}y : Wy \geq s, \ y \in \mathbb{Z}^m_+\}. \tag{3}$$

Here c is an n-dimensional vector, $C = \{x \in \mathbb{R}^n_+ : Ax \geq b\}$ is a nonempty polyhedron with A a $q \times n$ matrix and b a q-dimensional vector, T is a $p \times n$-matrix, $\tilde{q}$ is an m-dimensional vector, and W a $p \times m$-matrix. All matrices/vectors have real elements except for W, which is a rational matrix. ξ is a random vector in $\mathbb{R}^p$ and E_ξ denotes the expectation with respect to ξ. For the moment, we assume that both Q and v are well-defined; conditions ensuring that will be given later on.

The stochastic program (1) is designed to finding optimal first-stage decisions x in an optimization problem under uncertainty where the first-stage decisions have

to be made before knowing the outcome of the random vector ξ, and where second-stage decisions y serve to compensate possible infeasibilities after having fixed x and observed ξ. For an introduction into two-stage stochastic programming we refer to [12].

In contrast to the many algorithms for stochastic programs with continuous second-stage variables (see e.g. [9]), up to now there is no method that can handle the above integer second stage efficiently. There are two main difficulties in solving stochastic integer programming problems. The first one is that, in order to compute one function value $Q(x)$, one has to solve many different (but similar) integer programs (which are in general $\mathcal{NP}$-hard).

The other difficulty is that the integer requirements on y lead to the value function v in (3) being in general only lower semicontinuous (see [3]) instead of convex for continuous variables y. This destroys the convexity of Q met in the non-integer case and solving (1) amounts to minimizing a non-convex and possibly discontinuous objective. The latter is a fairly recent field of research with promising first results [8].

In this paper, we present a framework for solving (1) where the second-stage integer programs are handled efficiently via Gröbner basis methods from computational algebra. Employing traditional stochastic programming methodology, the algebraic techniques are embedded into an algorithmic framework that reduces solving (1) to inspecting finite sets of candidate points.

Our approach is based on the following main ideas:
We solve the continuous relaxation (i.e. the stochastic program where the integer requirements in the second stage are dropped) to obtain rough initial information on the location of the optimal first-stage decisions of the stochastic integer program. These optimal decisions can be shown to belong to an explicit countable set, which, under mild assumptions, is even finite. The key problem of computing function values of Q (which involves solving the second-stage problem for various right-hand sides) is tackled via a method employing Buchberger's algorithm for computing Gröbner bases of polynomial ideals [6]. The integer program is translated into a subalgebra membership problem in a ring of polynomials. The latter is solved by an algorithm for the division with remainder of multivariate polynomials that generalizes the well-known division scheme for polynomials in one variable. A Gröbner basis of a certain polynomial ideal enters as the essential ingredient into that procedure. For the various right-hand sides in the initial integer program always the same Gröbner basis applies, such that solving the integer program for another right-hand side amounts to just another generalized division of multivariate polynomials. Computing the Gröbner basis is a hard job (in fact, the bottleneck of our method) but only has to be done once. Moreover, for the Gröbner basis computation only algebraic information contained in the second stage is needed, such that the Gröbner basis does not depend on the distribution of the random vector ξ.

Applying the above method to integer linear programs has been proposed for the first time in [6], (see also [26]). It yields additional information at a possibly high computational cost that can make it an inefficient method to solve the problem for a fixed right-hand side. However, the additional information turns out most beneficial when solving an integer linear program with varying right-hand side. Therefore, the above method does seem particularly useful for solving integer recourse stochastic programs.

The main body of research on two-stage stochastic integer programming has been devoted to structural properties of the second-stage expected value function Q. In [21], [22], complete recourse models are treated, whereas in [13], [14], [17], the focus is on simple integer recourse. In the latter case, explicit expressions for the expected value function are known. Moreover, if only the right-hand side vector is random and follows a finite discrete distribution, then the model can be translated

into a continuous simple recourse model by computation of the convex hull.

First attempts to solve stochastic integer programs are described in [15] and [16]. The former approaches the problem by dynamic programming, and can be used only for problems of relatively small size. The approach in the latter paper is based on the assumption that the first stage variables are also integral, thereby obtaining countability of the set of feasible solutions. Moreover, it is implicitly assumed that computation of the second stage integer problems provides no hardship.

Inspired by a discussion of our basic ideas, Tayur [25] applied Gröbner basis methods to approximate objective function values for a particular plant management problem.

In Section 2 we describe how a Gröbner basis can be used to evaluate the objective function of a stochastic integer program. The computational bottleneck here is to find the Gröbner basis. Once having it, function evaluations are cheap.

In Section 3 we show that, if the random vector ξ is discretely distributed as we will assume, an optimal solution of (1) is contained in a certain countable set. Under some mild conditions this set of points can even be reduced to a finite set, as exposed in Section 4.

For an effective method the above set of points is to be enumerated completely. In Section 5 we propose a basic enumeration scheme that does this job.

Having presented all ingredients, they are put together in Section 6 which contains a short description of the algorithm. This is followed by two possible improvements, both directed at reducing the number of points to be evaluated .

Conclusions and directions for future research on this topic are presented in Section 7.

Finally, a short introduction into Gröbner basis methods for integer programs with various references to the literature can be found in the Appendix.

2 Function evaluations using Gröbner bases

Solving the integer programs behind the function values of the expected recourse function Q is the key problem in integer recourse stochastic programming. Here, we have to solve

$$\min\{\tilde{q}y : Wy \geq s,\ y \in \mathbb{Z}_+^m\}$$

for arbitrary right-hand sides $s \in \mathbb{Z}^p$ (we assume that W is an integer matrix, hence if $s \notin \mathbb{Z}^p$ then it can be replaced by its componentwise integer round up $\lceil s \rceil$). Without loss of generality, we can assume that, after introducing slack variables, the problem (with properly adjusted $\tilde{q}, W$) is in equality form

$$\text{(P)} \quad \min\{\tilde{q}\mathsf{y} : W\mathsf{y} = s,\ \mathsf{y} \in \mathbb{Z}_+^{\bar{m}}\}$$

with $\bar{m} = m + p$.

For problems of this type, recently, a solution technique was developed that is based on Gröbner basis methods from computational algebra [6]. It will turn out that this method is particularly adapted to our needs, since it allows a considerable short cut when resolving (a moderately sized) problem (P) for varying right-hand side. Below we apply the method to (P) and give short explanations of what is behind the single steps. Further details on Gröbner basis methods for integer programs can be found in the Appendix.

Buchberger Algorithm for Solving the Second-Stage Programs

STEP 1: Let k be any field and fix the polynomial ring $k[\mathsf{x}_1, \ldots, \mathsf{x}_p, \mathsf{y}_1, \ldots, \mathsf{y}_{\bar{m}}, \mathsf{t}]$.

STEP 2: Specify a monomial order $\prec$ in $k[\mathsf{x},\mathsf{y},\mathsf{t}]$ that is compatible with $\tilde{q}$ and guarantees $\mathsf{x} > \mathsf{y}, \mathsf{t} > \mathsf{y}$.

STEP 3: Form the ideal $I = \langle \mathsf{x}^{w^{1'}} - \mathsf{x}^{w^{1''}}\mathsf{y}_1, \ldots, \mathsf{x}^{w^{\bar{m}'}} - \mathsf{x}^{w^{\bar{m}''}}\mathsf{y}_{\bar{m}}, \mathsf{x}_1 \cdot \ldots \cdot \mathsf{x}_p \cdot \mathsf{t} - 1\rangle$.

STEP 4: Compute the (reduced) Gröbner basis G of I with respect to $\prec$ using Buchberger's algorithm.

STEP 5: Divide x^s by G, yielding the remainder $r_G(\mathsf{x}^s)$.

STEP 6: If $r_G(\mathsf{x}^s) \in k[\mathsf{y}]$ then the remainder is a monomial whose exponent vector is an optimal solution to (P), otherwise, (P) has no feasible solution.

The method's basic feature is to translate linear diophantine equations into relations in rings of polynomials and to treat problems like solving an equation or minimizing over the solution set to an equation via computations involving a generalized division scheme for multivariate polynomials. Integer vectors typically reappear as exponent vectors of multivariate monomials. The crucial part of the method is STEP 4 where a powerful tool from computational algebra, Buchberger's algorithm, is used.

In STEP 1, the underlying ring of polynomials is selected. We have a $p-$vector x of indeterminates corresponding to the number of rows in W and a $\bar{m}-$vector y corresponding to the number of columns (the number of variables in (P)). The variable t has a supporting function that will be explained in STEP 3.

In STEP 2, the objective function $\tilde{q}$ is incorporated by fixing a total order of the (exponents of the) monomials in $k[\mathsf{x},\mathsf{y},\mathsf{t}]$. Compatibility with $\tilde{q}$ means that monomials in y are ordered in the same way as their exponent vectors are ordered by the objective $\tilde{q}$. The notation $\mathsf{x} > \mathsf{y}$ (and, analogously, $\mathsf{t} > \mathsf{y}$) reflects that any monomial containing a nontrivial x-component dominates any monomial in y.

In STEP 3, an ideal in $k[\mathsf{x},\mathsf{y},\mathsf{t}]$ is defined via a set of generators. By the first $\bar{m}$ generators, the matrix W is translated. Each column $w^l \quad (l = 1,\ldots,\bar{m})$ is represented as $w^l = w^{l'} - w^{l''}$ with $w^{l''} = w^l_o\mathbf{1}$ where $w^{l'}, w^{l''} \in \mathbb{Z}^p_+, w^l_o \in \mathbb{Z}_+$ and $\mathbf{1}$ denotes the vector of all ones. This representation being non-unique equivalence classes have to be singled out which is done by the last generator containing the variable t.

In STEP 4, the tremendous part of the work is done: Another generating set of I with favorable properties, the so-called reduced Gröbner basis G, is computed by the Buchberger algorithm. The basic computation in Buchberger's algorithm, repeated a huge number of times, is a generalized division with remainder of a multivariate polynomial by a set of multivariate polynomials. It generalizes the well-known division scheme for polynomials in one variable.

STEPS 5 and 6: Here, the problem (P) is actually solved. In the background, there is a translation of (P) into the problem whether the polynomial x^s belongs to a certain subalgebra in $k[\mathsf{x},\mathsf{y},\mathsf{t}]$ [6]. The latter is decided constructively using a result in [23]: One computes the remainder $r_G(\mathsf{x}^s)$ of x^s on division by G. Here, the remainder is a monomial. If it is in $k[\mathsf{y}]$ then the exponent vector solves (P), otherwise, (P) has no feasible solution.

Computing Gröbner bases via Buchberger's algorithm is of exponential complexity and solving large instances of (P) by the above method is far from today's possibilities. However, for problems with moderate size Gröbner bases can be found. They contain the essential information to organize efficiently the repeated solution of (P) for varying right-hand side. To our knowledge, no other method can supply comparable information.

Note that the right-hand side s only enters in STEP 5, i.e., after computing the Gröbner basis. Solving (P) then amounts to a single generalized division whereas computing G can involve millions of such divisions. Therefore, having once computed the (reduced) Gröbner basis corresponding to (P), the computation of the value function

$$v(s) \;=\; \min\{\tilde{q}y : Wy \geq s,\; y \in \mathbb{Z}_+^m\}$$

is cheap!

To illustrate that and without aiming at an efficient implementation we enclose some initial insight into the computational behavior of the method. To carry out STEPS 4 and 5 we used the general purpose computer algebra package CoCoA [10]. Present research focuses on exploiting the structure of the binomial ideal I. In this context, we refer to a geometric interpretation in [26] and a very recent implementation reported in [11].

We ran our tests on knapsack problems. Of course, the latter have to be fitted into the form (P) to apply the above method. This leads to additional variables and equality constraints with direct impact on the number of indeterminates in the ring $k[\mathsf{x}, \mathsf{y}]$. To avoid technicalities, problems are listed in their usual form.

Note that, here, W has only non-negative entries such that it is possible to put $w^{l''} = 0 \quad (l = 1, \ldots, \bar{m})$. Then, the variable t is not needed and I reads $I = \langle \mathsf{x}^{w^1} - \mathsf{y}_1, \ldots, \mathsf{x}^{w^{\bar{m}}} - \mathsf{y}_{\bar{m}} \rangle$.

Computations were made on a 486DX4/100 PC.

Example 2.1

(i) $\max\{16\mathsf{y}_1 + 19\mathsf{y}_2 + 23\mathsf{y}_3 + 28\mathsf{y}_4 \;:$
$2\mathsf{y}_1 + 3\mathsf{y}_2 + 4\mathsf{y}_3 + 5\mathsf{y}_4 \leq s,\; \mathsf{y}_l \in \{0,1\}, l = 1, \ldots, 4\}$

The underlying ring $k[\mathsf{x}, \mathsf{y}]$ has 14 indeterminates. A reduced Gröbner basis with 45 elements was found in 1.1 seconds.

(ii) $\max\{16\mathsf{y}_1 + 19\mathsf{y}_2 + 23\mathsf{y}_3 + 28\mathsf{y}_4 \;:$
$2\mathsf{y}_1 + 3\mathsf{y}_2 + 4\mathsf{y}_3 + 5\mathsf{y}_4 \leq s_1,$
$6\mathsf{y}_1 + \;\mathsf{y}_2 + 3\mathsf{y}_3 + 2\mathsf{y}_4 \leq s_2,\; \mathsf{y}_l \in \{0,1\}, l = 1, \ldots, 4\}$

The underlying ring $k[\mathsf{x}, \mathsf{y}]$ has 16 indeterminates. A reduced Gröbner basis with 55 elements was found in 2.2 seconds.

(iii) $\max\{16\mathsf{y}_1 + 19\mathsf{y}_2 + 23\mathsf{y}_3 + 28\mathsf{y}_4 + 32\mathsf{y}_5 + 35\mathsf{y}_6 + 40\mathsf{y}_7 \;:$
$2\mathsf{y}_1 + 3\mathsf{y}_2 + 4\mathsf{y}_3 + 5\mathsf{y}_4 + 6\mathsf{y}_5 + 7\mathsf{y}_6 + 8\mathsf{y}_7 \leq s,\; \mathsf{y}_l \in \{0,1\}, l = 1, \ldots, 7\}$

The underlying ring $k[\mathsf{x}, \mathsf{y}]$ has 23 indeterminates (the maximum possible in the CoCoA version we had). A reduced Gröbner basis with 471 elements was found in 179.2 seconds.

□

After having computed the Gröbner bases we solved the above knapsack problems for various right-hand sides s by carrying out the division in STEP 5 of the above method. For the first two problems the computing times were at most 0.06 seconds, for the last problem they ranged from 0.05 to 0.17 seconds.

3 A countable number of function evaluations

In this section we show that the set of points in which evaluation of the objective function by the method presented in the previous section is required is countable in case right-hand sides ξ follow a discrete distribution. We first give general assumptions, also known from continuous recourse modelling, that assure that our model is well defined. Then we present additional assumptions and resulting properties of the function Q and the model (1), that lead to countability of the number of operations required by our method.

3.1 Assumptions

We assume that

(i) For any $s \in \mathbb{R}^p$ there exists a $y \in \mathbb{Z}^m_+$ such that $Wy \geq s$.

(ii) There exists a $u \in \mathbb{R}^p_+$ such that $W^T u \leq \tilde{q}$.

(iii) The random vector ξ has finite first moment.

Assumption (i) says that, for any possible value of $Tx - \xi$, there exists a feasible second-stage (or recourse) decision y. Following the continuous-recourse terminology, we say that (1) has complete integer recourse. By assumption (ii), the dual to the continuous relaxation of (3) has a feasible point. Therefore, (i) and (ii) together imply that $v(Tx - \xi) \in \mathbb{R}$, for all $x \in \mathbb{R}^n$ and all $\xi \in \mathbb{R}^p$ (Proposition I.6.7. in [18]). Moreover, there exist constants $a_1, a_2 \in \mathbb{R}$ such that for all $s_1, s_2 \in \mathbb{R}^p$

$$|v(s_1) - v(s_2)| \leq a_1 ||s_1 - s_2|| + a_2$$

(Theorem 8.1, [1]; Theorem 2.1, [3]). Therefore, assumptions (i) - (iii) imply that $Q(x) \in \mathbb{R}$ for all $x \in \mathbb{R}^n$, and (1) is well-defined.

The model (1) is a special case of the mixed-integer recourse model studied in [22]. As a consequence of Proposition 3.1 in [22] we obtain

Lemma 3.1 Assume (i) - (iii), then Q is a real-valued lower semicontinuous function on $\mathbb{R}^n$, i.e. $\liminf_{x \to x_o} Q(x) \geq Q(x_o)$ for all $x_o \in \mathbb{R}^n$.

We assume throughout the paper:

(iv) The random vector ξ follows a discrete distribution with finite support Ξ, say $\Xi = \{\xi^1, \xi^2, \ldots \xi^r\}$ and $p^i = Pr(\xi = \xi^i)$.

It is shown in [22] that, under mild assumptions, local optimal values and sets of local optimal solutions to (1) behave stable if the distribution of ξ is perturbed with respect to the topology of weak convergence of probability measures (Proposition 4.1, [22]). Therefore, it is possible to resort to discrete distributions of ξ when solving (1). If ξ has a continuous distribution then, according to the mentioned stability result, solutions to (1) can be approximated with any given accuracy if a discrete distribution is taken for ξ that is sufficiently close to the original one in the topology of weak convergence.

To be able to apply the Gröbner basis algorithm for function evaluations we assume:

(v) All elements of W are integers.

Actually, it is sufficient if W is rational. Integrality is then obtained by scaling.

Moreover, assumption (v) serves to facilitate specification of the sets where the function Q is constant.

3.2 Countability

To establish countability of the number of function evaluations required under the above assumptions, we first analyze the structure of the expected value function. In the following, $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ denote (componentwise) integer round up and round down, respectively.

For all non-negative integer vectors y, $Wy \geq t$ implies $Wy \geq \lceil t \rceil$. Therefore, the second-stage value function v is constant on subsets

$$\{s \in \mathbb{R}^p : \lceil s \rceil = k\} = \{s : k - (1, \ldots, 1)' < s \leq k\} \qquad \forall k \in \mathbb{Z}^p,$$

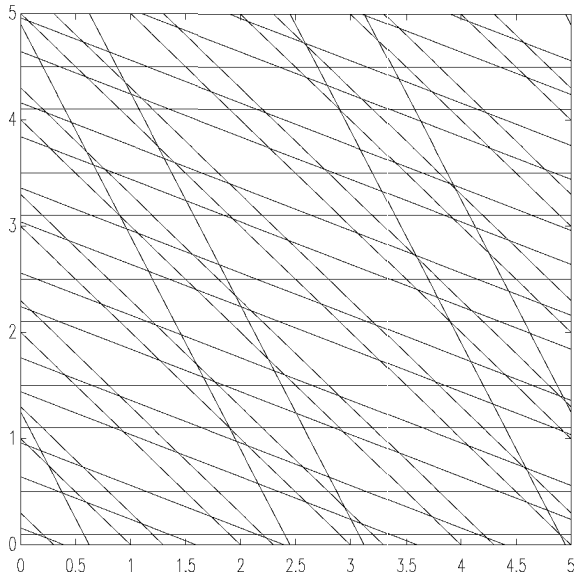

Figure 1: Example of the partition of $[0,5] \times [0,5]$ in sets $C(\cdot)$ if T and Ξ are given by (5).

and the function Q is constant on intersections of such subsets. For every $\bar{x} \in \mathbb{R}^n$, the function Q is constant on

$$\begin{aligned} C(\bar{x}) &= \bigcap_{i=1}^{r} \left\{x : \lceil Tx - \xi^i \rceil = \lceil T\bar{x} - \xi^i \rceil\right\} \\ &= \bigcap_{i=1}^{r} \bigcap_{j=1}^{p} \left\{x : \lceil T_j x - \xi_j^i \rceil = \lceil T_j \bar{x} - \xi_j^i \rceil\right\} \\ &= \bigcap_{i=1}^{r} \bigcap_{j=1}^{p} \left\{x : \lceil T_j \bar{x} - \xi_j^i \rceil + \xi_j^i - 1 < T_j x \leq \lceil T_j \bar{x} - \xi_j^i \rceil + \xi_j^i\right\}. \end{aligned} \tag{4}$$

Here T_j is the jth row of the matrix T and ξ_j^i is the jth component of the ith vector in the support of ξ.

From (4) we see that every set $C(\cdot)$ is obtained by intersecting $r \times p$ sets of the form $k_j^i + \langle \xi_j^i \rangle - 1 < T_j x \leq k_j^i + \langle \xi_j^i \rangle$, where $k_j^i \in \mathbb{Z}$ and $\langle \xi_j^i \rangle = \xi_j^i - \lfloor \xi_j^i \rfloor$ is the fractional part of ξ_j^i. Using this structure, in principle we can construct a partition of the feasible set $C = \{x \in \mathbb{R}_+^n : Ax \geq b\}$ in sets where the expected value function Q is constant. See Figure 1 for an example with

$$T = \begin{pmatrix} .4 & .2 \\ 1 & 1 \\ .5 & 1.25 \\ 0 & 1 \end{pmatrix} \qquad \Xi = \left\{ \begin{pmatrix} .25 \\ .3 \\ .2 \\ .1 \end{pmatrix}, \begin{pmatrix} .98 \\ 0 \\ .8 \\ .5 \end{pmatrix} \right\}. \tag{5}$$

Notice that, since each of the constituting sets is the intersection of an open and a closed half-space, in general the sets $C(\cdot)$ are neither open nor closed.

Below we will show how the fact that Q is constant on every set $C(\cdot) \cap C$ can be used to locate an optimal solution of (1), at least if such sets have vertices. The following condition guarantees this. According to [19], § 8 we denote by 0^+C the recession cone of the convex polyhedron C, i.e. the set of all directions $w \in \mathbb{R}^n$ such that $x + tw \in C$ for some $x \in C$ and all $t \geq 0$. If

$$0^+C \cap \{x : Tx = 0\} = \{0\} \tag{6}$$

then each of the sets $C(\cdot) \cap C$ is bounded and has vertices.

In the next section we will show that, under a mild additional assumption, we can restrict the search for optimal solutions to a bounded set. In that case existence of vertices is no longer a question and the condition (6) can be dropped.

In the following theorem we show that the countable set of all vertices of the sets $C(\cdot) \cap C$ contains an optimal solution of (1).

Definition 3.1 The countable set V, given by

$$V = \{x \in \mathbb{R}^n : x \text{ is a vertex of } C(x) \cap C\},$$

is called the *set of candidates*; an element of V is called a *candidate point*.

Theorem 3.1 Let V, the set of candidates, be non-empty. If $\operatorname{argmin}\{cx + Q(x) : x \in C\} \neq \emptyset$ then

$$V \cap \operatorname{argmin}\{cx + Q(x) : x \in C\} \neq \emptyset.$$

PROOF. Let $\bar{x} \in \operatorname{argmin}\{cx + Q(x) : x \in C\}$. For all $x \in C(\bar{x}) \cap C$ we have $Q(x) = Q(\bar{x})$. Consider minimizing the linear function $cx + Q(\bar{x})$ on the closure of $C(\bar{x}) \cap C$, denoted by $\operatorname{cl}(C(\bar{x}) \cap C)$. Since the minimum over this set is attained, it is attained in one of its vertices, say $\hat{x}$. If $\hat{x} \in C(\bar{x}) \cap C$ we are finished, since in that case $Q(\hat{x}) = Q(\bar{x})$ and $c\hat{x} \leq c\bar{x}$, implying that the vertex $\hat{x}$ is an optimal solution. Otherwise, i.e. if $\hat{x} \in \operatorname{cl}(C(\bar{x}) \cap C) \setminus C(\bar{x}) \cap C$, consider the set $C(\hat{x}) \cap C$ which trivially contains $\hat{x}$ as a vertex. It holds

$$Q(\hat{x}) \leq \lim_{x \to \hat{x}} Q(x) = Q(\bar{x}) \qquad \forall x \in C(\bar{x}) \cap C,$$

where the inequality is valid by the lower semicontinuity of Q. Since also $c\hat{x} \leq c\bar{x}$, it follows that $\hat{x} \in V$ is an optimal solution. □

Thus, in order to find an optimal solution of (1), it is sufficient to consider only elements of the countable set V. In the next section we present conditions such that the set of candidates is finite.

4 Finiteness using the continuous relaxation

The purpose of this section is to show how the continuous relaxation of (1) can be used to reduce the set of candidates defined in the previous section to a finite set, under mild conditions. The continuous relaxation is obtained by dropping the integrality conditions on the second-stage variables in (3):

$$\min\{cx + Q_R(x) \ : \ x \in C\} \tag{7}$$

where

$$Q_R(x) = E_\xi \, v_R(Tx - \xi) \tag{8}$$

and

$$v_R(s) = \min\{\tilde{q}y : Wy \geq s, \ y \in \mathbb{R}^m_+\}. \tag{9}$$

By (i) - (iii), the problem (7) is well defined. Its optimal value is, clearly, a lower bound to the optimal value of (1).

The following result is the basic tool that allows the use of the continuous relaxation to restrict the set of candidate points.

Lemma 4.1 Let C be a non-empty set, and f and $\bar{f}$ real functions on C such that $\bar{f}(x) \leq f(x)$ for all $x \in C$. Then, for all $\bar{x} \in C$,

$$\operatorname*{argmin}_{x \in C} f(x) \subset \{x \in C : \bar{f}(x) \leq f(\bar{x})\}.$$

Moreover, the difference between these sets is smaller according as $\bar{f}$ is a better approximation of f and $f(\bar{x})$ is a better approximation of $\inf_{x \in C} f(x)$. In particular, if $\bar{f}(\bar{x}) = f(\bar{x})$ and $\bar{x} \in \operatorname{argmin}_{x \in C} \bar{f}(x)$ then $\bar{x} \in \operatorname{argmin}_{x \in C} f(x)$.

Proof. For any $\bar{x} \in C$

$$\begin{aligned}
\operatorname*{argmin}_{x \in C} f(x) &= \bigcap_{y \in C} \{x \in C : f(x) \leq f(y)\} \\
&\subset \bigcap_{y \in C} \{x \in C : \bar{f}(x) \leq f(y)\} \\
&\subset \{x \in C : \bar{f}(x) \leq f(\bar{x})\},
\end{aligned}$$

where the tightness of each inclusion clearly depends on the indicated properties of f and $\bar{x}$, respectively.

The last claim follows directly from the assumptions. We have $f(\bar{x}) = \bar{f}(\bar{x}) \leq \bar{f}(x) \leq f(x)$ for all $x \in C$, which precisely means that $\bar{x} \in \operatorname{argmin}_{x \in C} f(x)$. □

Since Q_R is a lower bound for Q on $\mathbb{R}^n$, this lemma implies that for any feasible $\bar{x}$ the corresponding level set of the objective of the continuous relaxation, denoted by $L(c\bar{x}+Q(\bar{x}))$, contains all minimizers of the integer recourse problem (1). Moreover, each time a feasible point with a lower objective value is found, the level set can be shrinked, thus reducing the number of points that have to be enumerated even further.

It is immediately clear now that if there is a bounded level set then the set of candidates is finite. To arrive at conditions under which this is the case, we first review a well-known dual representation of the function Q_R and using this representation we discuss a (partial) description of the level sets $L(\cdot)$. We will use these also in the next section where we discuss how to actually enumerate the set of candidates.

By linear programming duality, we obtain

$$v_R(s) = \max\{su : W'u \leq \tilde{q}, u \in \mathbb{R}^p_+\}.$$

Assumptions (i) - (ii) together imply that the set $M_D = \{u \in \mathbb{R}^p_+ : W^T u \leq \tilde{q}\}$ is a nonempty compact polyhedron. Denoting its vertices by $d_1, \ldots, d_{N_o}$ we obtain

$$v_R(s) = \max_{l=1,\ldots,N_o} d_l s.$$

Hence,

$$Q_R(x) = \sum_{i=1}^{r} p^i v_R(Tx - \xi^i) = \sum_{i=1}^{r} p^i \max_{l=1,\ldots,N_o} d_l(Tx - \xi^i)$$

is a piecewise linear convex function on $\mathbb{R}^n$.

As explained above, approximations of lower level sets associated with Q_R are used in our algorithm. These sets are constructed as follows:

With a subset $\{d_1, \ldots, d_N\}$ $(N \leq N_o)$ of the vertices of M_D, the function

$$\bar{Q}_R(x) = \max_{l=1,\ldots,N} d_l(Tx - \bar{\xi})$$

where

$$\bar{\xi} = \sum_{i=1}^{r} p^i \xi^i$$

forms a lower bound to Q_R. Indeed, by the convexity of Q_R and Jensen's inequality, we have for all $x \in \mathbb{R}^n$

$$\begin{aligned} Q_R(x) &\geq v_R(Tx - \bar{\xi}) \\ &= \max_{l=1,\ldots,N_o} d_l(Tx - \bar{\xi}) \\ &\geq \max_{l=1,\ldots,N} d_l(Tx - \bar{\xi}) \\ &= \bar{Q}_R(x). \end{aligned}$$

An (outer) approximation of the lower level set

$$L(\alpha) = \{x \in C : cx + Q_R(x) \leq \alpha\} \tag{10}$$

is thus given by

$$\bar{L}(\alpha) = \{x \in C : cx + d_l(Tx - \bar{\xi}) \leq \alpha,\ l = 1, \ldots, N\}. \tag{11}$$

For problems with very moderate size, the complete list of vertices of M_D can be obtained via stochastic programming pre-processing techniques as in [12], [28], or by general vertex enumeration methods (cf. [5] for a comfortable implementation). In general, however, one has to live with a partial list of vertices. Algorithms for (non-integer) stochastic programs like the regularized decomposition method [20] yield such a list in the course of computation. Since, later on, we begin solving (1) by solving its continuous relaxation, we can assume that at least a partial list of vertices of M_D is available.

As mentioned before, for our set of candidates, and hence for our method, to be finite, it will be essential that $\bar{L}(\alpha)$ is bounded. Therefore, we add some simple conditions to enforce the latter. Recall that 0^+C denotes the recession cone of C.

Lemma 4.2 A nonempty approximate level set $\bar{L}(\alpha)$ is bounded, provided that

$$\{w \in 0^+C \ :\ (c + d_l T)w \leq 0,\ l = 1, \ldots, N\} = \{0\}.$$

Proof. Obviously,

$$\bar{L}(\alpha) \subseteq \{x \in C \ :\ cx + d_l Tx \leq \alpha + \max_l d_l \bar{\xi},\ l = 1, \ldots, N\}.$$

By our assumption, the recession cone of the latter set is $\{0\}$. Therefore, this set is bounded by Theorem 8.4 in [19]. □

Lemma 4.3 If the solution set to the continuous relaxation (7) is nonempty and bounded, then

$$\{w \in 0^+C \ :\ (c + d_l T)w \leq 0,\ l = 1, \ldots, N_o\} = \{0\}.$$

Proof. Assume that there exists $w \neq 0$ such that $w \in 0^+C$ and $(c + d_l T)w \leq 0,\ l = 1, \ldots, N_o$, and let $\bar{x} \in \mathbb{R}^n$ be in the solution set to the continuous relaxation. Then it holds for all $t \geq 0$

$$c(\bar{x} + tw) + d_l(T(\bar{x} + tw) - \xi^i) \leq c\bar{x} + d_l(T\bar{x} - \xi^i),\ \ i = 1, \ldots, r,\ l = 1, \ldots, N_o.$$

Taking the maximum over l and summing up over i yields

$$c(\bar{x} + tw) + Q_R(\bar{x} + tw) \ \leq\ c\bar{x} + Q_R(x) \quad \text{for all } t \geq 0.$$

By $w \in 0^+C$, it holds $\bar{x} + tw \in C$ for all $t \geq 0$. By $w \neq 0$, this implies unboundedness of the solution set to the continuous relaxation, a contradiction. □

Remark 4.1 Provided the continuous relaxation has a nonempty, bounded solution set, it is possible to achieve boundedness of $\bar{L}(\alpha)$ by finding sufficiently many vertices $d_1, \ldots, d_N$, in an extreme case, all the $d_1, \ldots, d_{N_o}$.

5 Enumerating the set of candidates

In the previous sections we have shown that the set of candidates V, intersected with a level set L of the continuous relaxation, contains an optimal solution of (1). To obtain an optimal solution we need to enumerate this set completely, which is finite by assumption. In this section we show how such a complete enumeration can be organized in a way that takes advantage of the structure of the set of candidates.

By definition every candidate point $v \in V$ is a vertex of $C(v) \cap C$, which can be represented as

$$\left\{ x \in \mathbb{R}^n : \begin{array}{l} k_j^i + \langle \xi_j^i \rangle - 1 < T_j x \leq k_j^i + \langle \xi_j^i \rangle,\ i = 1 \ldots r,\ j = 1 \ldots p \\ Ax \geq b, x \geq 0 \end{array} \right\}, \qquad (12)$$

for suitable choices of $k_j^i \in \mathbb{Z}$, $i = 1 \ldots r$, $j = 1 \ldots p$. Since v is a vertex of this set, it satisfies n independent inequalities from this system with equality.

Remark 5.1 Obviously, we only need to consider candidate points that are contained in the level set L. However, the inequalities defining L are not represented in (12), since their role differs from the inequalities in terms of the rows of T and A, and the non-negativities. Indeed, a vertex of $C(\cdot) \cap L$ that is not also a vertex of $C(\cdot) \cap C$ is not a candidate point as defined in Definition 3.1, and therefore need not be evaluated.

It is not difficult to see that by considering all choices for $k_j^i \in \mathbb{Z}$ such that $\{x : T_j x = k_j^i + \langle \xi_j^i \rangle\} \cap L$ is non-empty, and for every such choice considering all combinations of n independent equalities, we can obtain a complete list of all candidate points in L. This idea is the basis of our enumeration method. However, for reasons that will be explained later on, our enumeration method uses the fact that V can be divided in subsets of candidate points that all lie on a line segment defined by $n-1$ out of n equality constraints as mentioned above. To give a detailed description of our enumeration method, we first need to introduce some notation.

For $j = 1, \ldots, p$, define

$$\begin{aligned} t_j^u &= \max\{T_j x : x \in L\} \\ t_j^l &= \min\{T_j x : x \in L\}, \end{aligned}$$

and

$$R_j = \bigcup_{i=1}^{r} \left\{ k + \langle \xi_j^i \rangle : k \in \mathbb{Z},\ t_j^l \leq k + \langle \xi_j^i \rangle \leq t_j^u \right\}.$$

Each set R_j contains all right-hand side values such that the inequality $T_j x \leq r_j$, $r_j \in R_j$, may appear in the description (12) of some set $C(\cdot)$ that has a non-empty intersection with L. Similarly, define $R_j = \{b_{j-p}\}$, $j = p+1, \ldots, p+q$, and $R_j = \{0\}$, $j = p+q+1, \ldots, p+q+n$, to denote the sets of possible right-hand side values for the inequalities defining the set $C = \{x \in \mathbb{R}^n : Ax \geq b, Ix \geq 0\}$, where I is the $n \times n$ identity matrix.

Define the $(p+q+n) \times n$ matrix $S = (T; A; I)$. Finally, let $J \subset \{1, \ldots, p+q+n\}$, $|J| = n-1$, be an index set such that the matrix S_J, consisting of the rows S_j, $j \in J$, has rank $n-1$, and let $R_J \subset \mathbb{R}^{n-1}$ be the cartesian product of R_j, $j \in J$.

For a fixed $r \in R_J$, consider

$$H_J(r) = \{x : S_J x = r\} \cap L.$$

Either $H_J(r) = \emptyset$, or it is a line segment with endpoints

$$\begin{aligned} x^u_J(r) &= \max\{cx : x \in H_J(r)\} \\ x^l_J(r) &= \min\{cx : x \in H_J(r)\} \end{aligned}$$

(if $c \perp H_J(r)$ use any other objective vector). Now consider a row S_j with $j \notin J$ (and S_j not perpendicular to $H_J(r)$). If, for some right-hand side $r_j \in R_j$, the hyperplane $S_j x = r_j$ intersects $H_J(r)$, then this intersection is a candidate point; we will say that this candidate point is generated on $H_J(r)$ by S_j. To obtain all candidate points on $H_J(r)$ generated by S_j we determine the intersections of $H_J(r)$ with all hyperplanes $S_j x = r_j$, where $r_j \in R_j$ only needs to be considered if r_j is in between $S_j x^u_J(r)$ and $S_j x^l_J(r)$. Repeating this procedure for every row S_j with $j \notin J$, and including $x^u_J(r)$ and/or $x^l_J(r)$ if it is on the boundary of the feasible set C, results in a list of all candidate points on $H_J(r)$.

By repeating the procedure above for every family of parallel line segments, i.e., for every possible subset J and every $r \in R_J$, all candidate points in L will be found. In fact, since every candidate point is on a line segments $H_J(\cdot)$ for n different sets J, it would be listed n times. This redundancy is easily removed by considering only candidates generated on $H_J(r)$ by rows S_j with $j > \max_J j$.

In Section 6.2 we will present some ideas to reduce the number of candidate points in which function evaluations are required. In particular, we will explain how to take advantage of the way we organized the enumeration.

6 Algorithm

We now have all but one of the ingredients that make up the algorithm to be presented in this section. The last ingredient concerns the determination of an appropriate (initial) level set. For this purpose we could choose any point in the feasible region C, evaluate the objective function in this point, and determine the level set using the continuous relaxation as described in the previous section. Since we need to solve the continuous relaxation to obtain a (partial) description of the level sets anyway, it seems reasonable to use one of its optimal solutions, say x^R, as initial point and $L(cx^R + Q(x^R))$ as the initial level set.

6.1 Basic form of the algorithm

The presentation of the algorithm is merely a summary of the ingredients exposed in the preceding sections. The algorithm consists of the following parts.

1. Compute a Gröbner basis for the second stage integer linear programming problem as explained in Section 2.
2. Solve the continuous relaxation (7). Let x^R be an optimal solution.
3. Compute the objective value $cx^R + Q(x^R)$ (using the Gröbner basis), and construct the (partial) level set $L(cx^R + Q(x^R))$.
4. For every candidate point in $L(cx^R + Q(x^R))$ evaluate the objective function, using the Gröbner basis to compute the expected value function Q. The candidate points are enumerated according to the scheme proposed in Section 5. A candidate with smallest function value is an optimal solution to the problem.

Under our assumption that the set $L(cx^R + Q(x^R))$ is bounded, the number of candidate points to be evaluated in step 4 is finite. Therefore, the algorithm determines an optimal solution of (1) in finite time.

6.2 Possible improvements

In this section we propose two possible improvements of the algorithm presented above. They both aim at reducing the number of candidate points for which the objective function has to be evaluated. In general, this number can be enormous. Moreover, every function evaluation takes r evaluations of the second stage value function v, where we recall that r is the number of mass points in the support Ξ. Thus, although each evaluation of v by means of the Gröbner basis is cheap, it is still worthwhile to try to minimize the number of function evaluations needed.

The first idea relates to the use of the level set L. By Lemma 4.1 all minimizers of (1) are contained in every level set $L(c\bar{x} + Q(\bar{x}))$ for every feasible solution $\bar{x}$. Since we only need to evaluate candidate points that are in the level set, it seems to be advantageous to choose the level set as small as possible. This can be implemented as follows. Recall that the initial level set is $L(cx^R + Q(x^R))$, where x^R is an optimal solution of the continuous relaxation (7). As before, we start evaluating function values in candidate points according to the enumeration scheme presented in Section 5. However, as soon as we find a candidate with a lower objective value than x^R, say $\hat{x}$, we can use it to shrink the level set to $L(c\hat{x} + Q(\hat{x}))$. This is of course a subset of $L(cx^R + Q(x^R))$, so that in general the number of remaining candidates is reduced. Clearly, this procedure can be repeated each time a lower function value is obtained.

Thus, repeatedly updating the level set has the benefit of reducing the number of candidates to be evaluated. However, this benefit should be set off against the additional work that the updating brings about. Updating the level set itself comes virtually free, since only the right-hand sides in (11) are changed. Considering the enumeration scheme, only the sets R_j, $j = 1, \ldots, p$, have to be updated. For each j, this boils down to recomputing t_j^u and t_j^l by solving an LP problem of size $(N+q)\times n$, where N is the number of inequalities in the polyhedral description of the (outer approximation of the) level set, and q is the number of rows of the matrix A. Our tentative conclusion is that it seems to be beneficial to update the level set, either every time a better solution is found or only if a significantly lower objective value is found. Numerical experiments would be useful to reveal the benefit of updating and may also give information on the best frequency of updating.

The second improvement makes use of the way we organized the complete enumeration as presented in Section 5, and actually is the main reason to choose this enumeration scheme.

Lemma 6.1 Let $H_J(r)$, $r \in R_J$, be a line segment with endpoints $x_J^l(r)$ and $x_J^u(r)$ as defined above. Let the candidate point v, not equal to $x_J^l(r)$ or $x_J^u(r)$, be generated on $H_J(r)$ only by rows T_j, $j \in I \subset \{1, \ldots, p\}$. Assume that

(i) $c(x_J^u(r) - x_J^l(r)) > 0$,

(ii) $T_j(x_J^u(r) - x_J^l(r)) > 0$ for all $j \in I$.

Then v is not an optimal solution of the integer recourse problem (1).

Proof. Let $\bar{v}$ be the neighboring candidate point of v on $H_J(r)$ in the direction $-(x_J^u(r) - x_J^l(r))$, and for an arbitrary $\lambda \in (0, 1)$ define $x_\lambda = \lambda\bar{v} + (1 - \lambda)v$ (see Figure 6.2). Obviously, $x_\lambda \in H_J(r)$. We will show that $x_\lambda \in C(v)$ so that $Q(x_\lambda) = Q(v)$. Since $cx_\lambda < cv$ by Assumption (i), this proves the result.

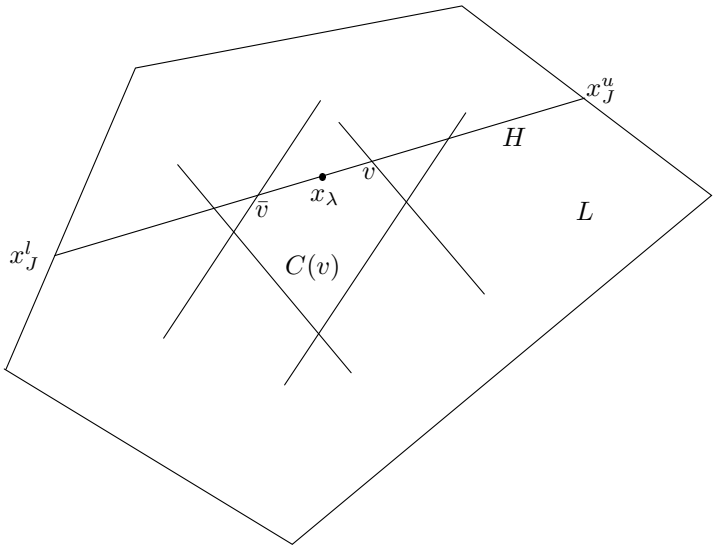

Figure 2: Illustration by the proof of Lemma 6.1.

Using (4) we have

$$C(v) = \bigcap_{j=1}^{p} C_j(v)$$

where

$$C_j(v) = \bigcap_{i=1}^{r} \left\{ x : \lceil T_j v - \xi_j^i \rceil - 1 < T_j x - \xi_j^i \leq \lceil T_j v - \xi_j^i \rceil \right\}.$$

Depending on the different roles that a row T_j may play, we distinguish three cases and each time show that $x_\lambda \in C_j(v)$.

If $j \in I$ then, since $x_\lambda \in H_J(r)$, it holds $T_j x_\lambda = T_j v$, so that $x_\lambda \in C_j(v)$.

If a row T_j generates v on $H_J(r)$ then, by assumption (ii), $T_j x_\lambda < T_j v$ so that $T_j x_\lambda - \xi_j^i \leq \lceil T_j v - \xi_j^i \rceil$ for all i. Since x_λ is in between the neighboring candidate points $\bar{v}$ and v we also have $\lceil T_j x_\lambda - \xi_j^i \rceil > \lceil T_j v - \xi_j^i \rceil - 1$ for all i, and we conclude that $x_\lambda \in C_j(v)$.

Finally, consider the case that $j \notin I$ and T_j does not generate v on $H_J(r)$. Then there are two possibilities: either $T_j v - \xi_j^i < \lceil T_j v - \xi_j^i \rceil$ for all i, or there exists an $\bar{i}$ such that $T_j v - \xi_j^{\bar{i}} = \lceil T_j v - \xi_j^{\bar{i}} \rceil$. In the first case it follows from the fact that there are no candidate points in between $\bar{v}$ and v that $x_\lambda \in C_j(v)$. In the latter case, since T_j does not generate v on $H_J(r)$, it must hold that $T_j x - \xi_j^{\bar{i}} = \lceil T_j v - \xi_j^{\bar{i}} \rceil$ for all $x \in H_J(r)$. Since $x_\lambda \in H_J(r)$ we have $T_j x_\lambda - \xi_j^{\bar{i}} = \lceil T_j v - \xi_j^{\bar{i}} \rceil$, so that also in this case $x_\lambda \in C_j(v)$. □

In our algorithm we may use Lemma 6.1 as follows. Consider a family of line segments $H_J(r)$, $r \in R_J$. If for some $r \in R_J$ it holds that $c(x_J^u(r) - x_J^l(r)) > 0$, then this is true for every $r \in R_J$ since all these line segments are parallel. Similarly, if a row T_j, $j \notin J$, satisfies $T_j(x_J^u(r) - x_J^l(r)) > 0$ for one $r \in R_J$, then this inequality holds for all $r \in R_J$. Therefore, given a family of line segments $H_J(r)$, $r \in R_J$, and a row T_j, $j \notin J$, we check for an arbitrary $r \in R_J$ if both $c(x_J^u(r) - x_J^l(r)) > 0$ and $T_j(x_J^u(r) - x_J^l(r)) > 0$. If these conditions are satisfied, we may skip all candidate

points that are generated by this row on any line segment $H_J(r)$, $r \in R_J$, since by Lemma 6.1 none of them can be optimal for (1) if they are generated only by this row T_j. Note that any candidate point that is also generated by a row T_i, $i \notin J$, $i \neq j$, such that $T_i(x^u_J(r) - x^l_J(r)) \leq 0$ will still be considered; hence, no possible optimal solutions will be discarded.

This improvement is very cheap to implement and may result in a significant reduction of the number of function evaluations needed.

7 Concluding remarks

The algorithm presented is one of the first general purpose algorithms devised for two-stage stochastic integer programming. The basic idea is that once having determined a Gröbner basis related to the matrix W and the objective $\tilde{q}$ the numerous solutions of the second stage integer program with varying right-hand sides take little time. Consequently, this technique for solving integer programs as proposed in [6] is preeminently suitable for solving stochastic integer programs. At the same time it also imposes limitations on the present applicability of the method: the present codes for determining a Gröbner basis can handle matrices of moderate size only. However, several research groups are working on special purpose variants of Buchberger's algorithm for solving (deterministic) integer programs, and any algorithmical progress in this field can be incorporated directly in our algorithm.

In addition to connecting Gröbner methodology with integer recourse problems we use structural properties of these problems to construct a countable c.q. finite set containing an optimal solution. As a result, the integer recourse problem can be solved using a (complete) enumeration scheme that is guided by the structure of the set of possible optimal solutions. It would be interesting to investigate the effect of the improvements proposed in Section 6.2 empirically and theoretically.

Appendix

Gröbner Bases

Let k be any field and $k[\mathtt{x}]$ denote the *ring of polynomials* in d variables $\mathtt{x} = (\mathtt{x}_1, \ldots, \mathtt{x}_d)$. Each *monomial* $\mathtt{x}^\alpha = \mathtt{x}_1^{\alpha_1} \cdot \mathtt{x}_2^{\alpha_2} \cdot \ldots \cdot \mathtt{x}_d^{\alpha_d}$ in $k[\mathtt{x}]$ is identified with its exponent vector α in the set $\mathbb{N}^d$ where $\mathbb{N}$ denotes the non-negative integers.

Using polynomials in the context of the present paper is basically motivated by the fact that linear equations with indeterminates in the non-negative integers have their counterparts in the language of polynomials. Indeed, given an equation $A\mathtt{y} = b$ where $\mathtt{y} \in \mathbb{N}^n$ is unknown and A, b are a non-negative integral $d \times n-$matrix and $d-$vector, respectively, we consider the monomials $\mathtt{x}^{a^1}, \ldots, \mathtt{x}^{a^n}$ and $\mathtt{x}^b$ in $k[\mathtt{x}]$ where $a^1, \ldots, a^n$ denote the columns of A. Then, there exists a solution $\mathtt{y} \in \mathbb{N}^n$ to $A\mathtt{y} = b$ if and only if $\mathtt{x}^b = (\mathtt{x}^{a^1})^{\mathtt{y}_1} \cdot \ldots \cdot (\mathtt{x}^{a^n})^{\mathtt{y}_n}$, i.e., if and only if the polynomial $\mathtt{x}^b$ can be written as a polynomial in $\mathtt{x}^{a^1}, \ldots, \mathtt{x}^{a^n}$. For the precise relations between integer linear programming and the theory of polynomials, however, it needs the algebraic prerequisites that we assemble in the present subsection.

A *monomial order* on $k[\mathtt{x}]$ is any relation $\prec$ on $\mathbb{N}^d$ that is a total order with 0 as the unique minimal element and with the property that $\alpha \prec \beta$ implies $\alpha + \gamma \prec \beta + \gamma$ for all $\alpha, \beta, \gamma \in \mathbb{N}^d$. A familiar example for a monomial order on $k[\mathtt{x}]$ is the (purely) lexicographic order, further examples are listed in [7]. For a polynomial $f \in k[\mathtt{x}]$, the maximal monomial with respect to a monomial order $\prec$ is called the *initial* (or *leading*) *monomial* and denoted by $in_\prec(f)$.

A subset $I \subset k[\mathbf{x}]$ is called an *ideal* if it satisfies: $0 \in I$, together with f_1, f_2 also their sum $f_1 + f_2$ belongs to I and if $f \in I$ and $h \in k[\mathbf{x}]$ then $hf \in k[\mathbf{x}]$. Given an arbitrary index set $\mathcal{I}$ and $f_i \in k[\mathbf{x}], i \in \mathcal{I}$, then the set $\langle f_i : i \in \mathcal{I}\rangle$ consisting of all polynomials which are finite sums of the form $\sum_{i\in\mathcal{I}} h_i f_i,\ h_i \in k[\mathbf{x}]$, is an ideal. We say that $I = \langle f_i : i \in \mathcal{I}\rangle$ is generated by $\{f_i : i \in \mathcal{I}\}$ and call the latter a *basis* of I. *Hilbert's Basis Theorem* asserts that every ideal $I \subset k[\mathbf{x}]$ has a finite basis. With an ideal I and a monomial order $\prec$ we define the *initial ideal* $in_\prec(I)$ by $in_\prec(I) = \langle in_\prec(f) : f \in I\rangle$.

Suppose you are given a finite basis $\{f_1, \ldots, f_\ell\}$ of an ideal I and a monomial order $\prec$. Of course, $\langle in_\prec(f_1), \ldots, in_\prec(f_\ell)\rangle \subseteq in_\prec(I)$, and this inclusion can be strict, in general (Example 2, [7], p. 74). A finite subset $G = \{g_1, \ldots, g_\ell\}$ of an ideal I is called a *Gröbner basis* of I with respect to a monomial order $\prec$ if $\langle in_\prec(f_1), \ldots, in_\prec(f_\ell)\rangle = in_\prec(I)$. It is possible to prove Hilbert's Basis Theorem in a way that simultaneously yields the existence of a Gröbner basis (Theorem 4, Corollary 6, [7], pp. 75,76). As another consequence it follows that I is generated by G.

If $k[\mathbf{x}]$ is equipped with a monomial order then the well-known algorithm for division with remainder of polynomials in one variable can be generalized to dividing a multivariate polynomial by a finite set of multivariate polynomials yielding a remainder. For details consult Chapter 2, § 3 in [7]. Given a monomial order $\prec$ and an (ordered) set $F = \{f_1, \ldots, f_\ell\} \subset k[\mathbf{x}]$, the *generalized division algorithm* rewrites $f \in k[\mathbf{x}]$ as $f = a_1 f_1 + a_2 f_2 + \ldots + a_\ell f_\ell + r_F(f)$ where $a_1, \ldots, a_\ell \in k[\mathbf{x}]$ and either $r_F(f) = 0$ or the remainder $r_F(f)$ is a $k-$linear combination of monomials none of which is divisible by any of $in_\prec(f_1), \ldots, in_\prec(f_\ell)$ (Theorem 3, [7], p. 63). In general, $r_F(f)$ depends on the order in which $f_1, \ldots, f_\ell$ are listed in F (Example 4, [7], p. 65). Moreover, it is clear that $r_F(f) = 0$ implies $f \in I = \langle in_\prec(f_1), \ldots, in_\prec(f_\ell)\rangle$. The reverse implication, however, can fail (Example 5, [7] p. 66). This is different if $G = \{g_1, \ldots, g_\ell\}$ is a Gröbner basis of an ideal $I \subset k[\mathbf{x}]$ with respect to a monomial order $\prec$. Then, for every $f \in k[\mathbf{x}]$, the remainder $r_G(f)$ of f on division by G is determined uniquely and $f \in I$ if and only if $r_G(f) = 0$.

An algorithm for computing Gröbner bases (Buchberger's Algorithm) is described in the next subsection. Gröbner bases together with Buchberger's Algorithm underlie many algorithms in algebraic geometry and commutative algebra, for details see [2], [7], [24].

Buchberger's Algorithm

To formulate a necessary and sufficient condition for a basis G of an ideal $I \subset k[\mathbf{x}]$ to be a Gröbner basis (with respect to a fixed monomial order $\prec$) we introduce the notion of an S-polynomial of two polynomials $f_1, f_2 \in k[\mathbf{x}]$. To avoid technicalities we assume that the coefficients (in k) of both $in_\prec(f_1)$ and $in_\prec(f_2)$ are equal to 1. Otherwise, the following can be adjusted in an obvious way. The *S-polynomial* $S(f_1, f_2)$ of f_1 and f_2 is defined by $S(f_1, f_2) = m_1 f_1 - m_2 f_2$ where m_1, m_2 are the unique monomials satisfying $m_1 in_\prec(f_1) = m_2 in_\prec(f_2)$ =least common multiple of $in_\prec(f_1)$ and $in_\prec(f_2)$, i.e., the S-polynomial is designed to yield cancellation of the leading monomials. The announced criterion now reads: A basis $G = \{g_1, \ldots, g_\ell\}$ of an ideal $I \subset k[\mathbf{x}]$ is a Gröbner basis if and only if for all pairs $i \neq j$, the remainder of $S(g_i, g_j)$ on division by G is zero (Theorem 6, [7], p. 84). This result is the key for constructing a Gröbner basis from an arbitrary finite basis of an ideal. The method is due to Buchberger and will be listed here only in its basic form. Many improvements were worked out by Buchberger and other authors (see [2], [4], [7] and the references therein).

Buchberger's Algorithm

Let $I = \langle f_1, \ldots, f_\ell\rangle$.

INPUT: $F = \{f_1, \ldots, f_\ell\}$
$G := F$
REPEAT $G' := G$
FOR each pair $\{p, q\}, p \neq q$, in G'
DO $S := r_{G'}(S(p, q))$
IF $S \neq 0$ THEN $G := G \cup \{S\}$
UNTIL $G = G'$.

OUTPUT: A Gröbner basis $G = \{g_1, \ldots, g_t\}$ of I with $F \subset G$.

Finiteness of the above algorithm essentially relies on the fact that after each passing through the main loop we have $\langle in_\prec(G')\rangle \subseteq \langle in_\prec(G)\rangle$, that this inclusion is strict if $G' \neq G$ and that there is no infinite strictly ascending chain of ideals in $k[\mathbf{x}]$ (Theorem 2, [7], p. 89).

Since Gröbner bases computed by the above algorithm are often unnecessarily big, unneeded generators are eliminated by the following two-step procedure. For the justification we refer to the arguments given at pp. 90,91 in [7].

STEP 1: FOR each $g \in G$
DO
IF there exists a $p \in G \setminus \{g\}$ such that $in_\prec(p)$ divides $in_\prec(g)$
THEN remove g from G

STEP 2: FOR each $g \in G$
DO $g := r_{G\setminus\{g\}}(g)$

Carrying out this procedure and normalizing the elements in G such that all the $k-$coefficients of their initial monomials are equal to 1 yields what is called the *reduced Gröbner basis*. If $\{0\} \neq I \subset k[\mathbf{x}]$ is an ideal then, for a given monomial order, I has a unique reduced Gröbner basis (Proposition 6, [7], p. 91).

It has to be pointed out that (reduced) Gröbner bases can have a large number of elements and that (minimal) bounds of the degrees in the members of a Gröbner basis can be quite large as well. In the literature, examples are given where the construction of a Gröbner basis for an ideal generated by polynomials of degree less than or equal to some d can involve polynomials of degree proportional to 2^{2^d} (see the discussion at the end of Chapter 2 in [7] and the references therein).

Buchberger's Algorithm for Integer Programs

Suppose we are given the integer linear program

$$\text{(P)} \qquad \min\{c\mathbf{y} \ : \ A\mathbf{y} = b, \mathbf{y} \in \mathbb{N}^n\}$$

where A is an integral $d \times n$-matrix and c, b are integral vectors of dimensions n and d, respectively. Recall that above the notion of a monomial order in a ring of polynomials was defined by referring to certain total orders of lattice points. The ranking induced by the objective-function vector c in (P) is now included into a monomial order on $\mathbb{N}^n$ by saying that the order is compatible with c. More precisely, a monomial order $\prec$ on $\mathbb{N}^n$ is compatible with $c \in \mathbb{Z}^n$ if $\alpha \prec \beta$ is equivalent to the alternative $c\alpha < c\beta$ or $c\alpha = c\beta$ and $\alpha \prec \beta$.

In [6] it is shown that, if (P) has a minimal solution, then a monomial order compatible to c exists. A possibility to specify monomial orders is to fix a suitable matrix B and to say that $\alpha \prec \beta$ if and only if $B\alpha$ is less than $B\beta$ lexicographically.

Now assume that c, A, b have non-negative entries. We will solve (P) by purely algebraic means. Although the same procedure, in principle, also applies for problems (P) with general integral data, in that case some technical extensions are

necessary to which we will come back later on. As sketched at the beginning of this appendix, the feasibility issue in (P) is equivalent to $\mathtt{x}^b$ being expressible as a polynomial in $\mathtt{x}^{a^1}, \ldots, \mathtt{x}^{a^n}$ where $a^1, \ldots, a^n$ are the columns of A. This, in fact, amounts to solving the subalgebra membership problem in $k[\mathtt{x}]$ whether $\mathtt{x}^b$ belongs to the subalgebra generated by $\mathtt{x}^{a^1}, \ldots, \mathtt{x}^{a^n}$.

Shannon and Sweedler [23] tackled subalgebra membership problems in $k[\mathtt{x}]$ via Gröbner bases. To outline their method let $k[\mathtt{x}] = k[\mathtt{x}_1, \ldots, \mathtt{x}_d], k[\mathtt{x}, \mathtt{y}] = k[\mathtt{x}_1, \ldots, \mathtt{x}_d, \mathtt{y}_1, \ldots, \mathtt{y}_n]$ be polynomial rings and $f_1, \ldots, f_n \in k[\mathtt{x}]$. Adopt a monomial order $\prec$ in $k[\mathtt{x}, \mathtt{y}]$ such that $\mathtt{x} > \mathtt{y}$, i.e., any monomial containing a non-trivial $\mathtt{x}$-component dominates any monomial in $\mathtt{y}$. Consider the ideal $I = \langle f_1 - \mathtt{y}_1, \ldots, f_n - \mathtt{y}_n \rangle \subset k[\mathtt{x}, \mathtt{y}]$ and let G be a Gröbner basis of I with respect to $\prec$. Then, in [23] it is shown that g belongs to the subalgebra generated by $f_1, \ldots, f_n$, i.e. $g \in k[f_1, \ldots, f_n]$, if and only if the remainder $r_G(g)$ of g on division by G belongs to $k[\mathtt{y}] = k[\mathtt{y}_1, \ldots, \mathtt{y}_n]$, and in this case $g = r_G(g)(f_1, \ldots, f_n)$, i.e., g is obtained by substituting $\mathtt{y}_1, \ldots, \mathtt{y}_n$ in $r_G(g)$ with $f_1, \ldots, f_n$.

Using the above membership algorithm Conti and Traverso [6] devised the following solution method for (P):

Buchberger Algorithm for Integer Programs

STEP 1: Design a ring $k[\mathtt{x}, \mathtt{y}]$ according to the dimension of A.
STEP 2: Specify a monomial order $\prec$ in $k[\mathtt{x}, \mathtt{y}]$ that is compatible with c and guarantees $\mathtt{x} > \mathtt{y}$.
STEP 3: Form the ideal $I = \langle \mathtt{x}^{a^1} - \mathtt{y}_1, \ldots, \mathtt{x}^{a^n} - \mathtt{y}_n \rangle$.
STEP 4: Compute the (reduced) Gröbner basis G of I with respect to $\prec$ using Buchberger's algorithm.
STEP 5: Divide $\mathtt{x}^b$ by G.
STEP 6: If $r_G(\mathtt{x}^b) \in k[\mathtt{y}]$ then the remainder is a monomial whose exponent vector is an optimal solution to (P), otherwise, (P) has no feasible solution.

Negative entries in A are handled in a way that, although in a different context, reminds to what is done in linear programming: each vector $a \in \mathbb{Z}^d$ can be written (not uniquely !) as $a = a' - a''$ where a', a'' have non-negative entries, $a'' = a_o \mathbf{1}$ with a non-negative integer a_o and $\mathbf{1}$ denoting the vector of all ones. In this way, one gets representations $a^{1'} - a^{1''}, \ldots, a^{n'} - a^{n''}$ of the columns of A and the above algorithm is modified as follows:

In STEP 1, another indeterminate $\mathtt{t}$ is introduced yielding the ring $k[\mathtt{x}, \mathtt{y}, \mathtt{t}]$. The monomial order in STEP 2 in addition has to satisfy $\mathtt{t} > \mathtt{y}$. The ideal in STEP 3 reads $I = \langle \mathtt{x}^{a^{1'}} - \mathtt{x}^{a^{1''}} \mathtt{y}_1, \ldots, \mathtt{x}^{a^{n'}} - \mathtt{x}^{a^{n''}} \mathtt{y}_n, \mathtt{x}_1 \cdot \ldots \cdot \mathtt{x}_d \cdot \mathtt{t} - 1 \rangle$. We only mention that the last binomial among the above generators serves to account for the non-uniqueness in representing the columns of A. For further details we refer to [6].

Let us finally add a few comments:

The tremendous part of the method is, of course, STEP 4. The exponential complexity of Buchberger's algorithm still prevents an application of the method to large-scale integer linear programs.

On the other hand, and this is crucial in the stochastic-programming context of the present paper, the right-hand side b enters only after the main part of the work is done: Having once computed the Gröbner basis, solving (P) for a specific b amounts to just a single generalized division with remainder.

In our situation, the input basis of I is given by binomials. Notice that the S-polynomial of two binomials is again a binomial and that dividing a binomial by a set of binomials yields as remainder either zero or a binomial. Therefore, G has to consist of binomials. Dividing $\mathtt{x}^b$, a monomial, by G, a set of binomials, hence yields a monomial as remainder.

In general, forming S-polynomials in the course of Buchberger's algorithm leads to polynomials that become longer and longer. In the present case, S-polynomials have length two which is, of course, beneficial when running Buchberger's algorithm.

Computer algebra packages usually contain implementations of Buchberger's algorithm. Therefore, the method outlined above can be implemented easily. One only has to make sure that the implementation at hand allows a free choice of the monomial order, which is impossible in some packages. However, the specifications hinted at above cannot be exploited with general purpose algorithms. Therefore, we refer to the work of Thomas [26], who gave a purely geometric interpretation of the method in terms of directed graphs whose knots are lattice points and whose edges at the beginning correspond to the input basis of I. Finding the reduced Gröbner basis via Buchberger's algorithm then means to rebuild the graph such that the edges correspond to the members of the Gröbner basis, which in turn can be understood as an integer programming test set. The geometric version offers the possibility of implementation via elementary manipulations of integer vectors.

As pointed out above the method involves a certain overhead when aiming at the solution of (P) for a single right-hand side b only. Based on the geometric insights of [26] recent research is directed to overcome this. First results along this line can be found in [27].

Acknowledgement

We wish to thank Rekha Thomas (Texas A&M University College Station) and Günter Ziegler (Technische Universität Berlin) for beneficial discussions.

References

[1] B. Bank and R. Mandel. *Parametric Integer Optimization*. Akademie-Verlag, Berlin, 1988.

[2] T. Becker and V. Weispfennig. *Gröbner Bases: A Computational Approach to Commutative Algebra*. Springer-Verlag, Berlin, 1993.

[3] C.E. Blair and R.G. Jeroslow. The value function of a mixed integer program: I. *Discrete Mathematics*, 19:121–138, 1977.

[4] B. Buchberger. Gröbner bases: An algorithmic method in polynomial ideal theory. In N.K. Bose, editor, *Multidimensional Systems Theory*, chapter 6. D. Reidel Publishing Company, 1985.

[5] T. Christof. PORTA - a polyhedron representation transformation algorithm. Available via `http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/PORTA/readme.html`.

[6] P. Conti and C. Traverso. Buchberger algorithm and integer programming. In *Proceedings AAECC-9 (New Orleans), Lecture Notes in Computer Science 539*, pages 130–139, Berlin, 1991. Springer-Verlag.

[7] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties and Algorithms*. Springer-Verlag, New York, 1992.

[8] Y.M. Ermoliev, V.I. Norkin, and R.J-B. Wets. The minimization of semicontinuous functions: mollifier subgradients. *SIAM Journal on Control and Optimization*, 33:149–167, 1995.

[9] Yu. Ermoliev and R.J-B. Wets. *Numerical Techniques for Stochastic Optimization*. Springer-Verlag, Berlin etc., 1988.

[10] A. Giovini, G. Niesi, and E. Armando. CoCoA: An experimental system for computer and commutative algebra, version 1.5. Department of Mathematics, University of Genova, 1991.

[11] S. Hosten and B. Sturmfels. GRIN: An implementation of Gröbner bases for integer programming. In E. Balas and J. Clausen, editors, *Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science 920*, pages 267–276. Springer-Verlag, Berlin, 1995.

[12] P. Kall and S.W. Wallace. *Stochastic Programming*. Wiley, Chichester etc., 1994.

[13] W.K. Klein Haneveld, L. Stougie, and M.H. van der Vlerk. On the convex hull of the simple integer recourse objective function. *Annals of Operations Research*, 56:209–224, 1995.

[14] W.K. Klein Haneveld and M.H Van der Vlerk. On the expected value function of a simple integer recourse problem with random technology matrix. *Journal of Computational and Applied Mathematics*, 56:45–53, 1994.

[15] B.J. Lageweg, J.K. Lenstra, A.H.G. Rinnooy Kan, and L. Stougie. Stochastic integer programming by dynamic programming. In Yu. Ermoliev and R.J-B Wets, editors, *Numerical Techniques for Stochastic Optimization*, chapter 21. Springer-Verlag, Berlin etc., 1988.

[16] G. Laporte and F.V. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13:133–142, 1993.

[17] F.V. Louveaux and M.H. Van der Vlerk. Stochastic programming with simple integer recourse. *Mathematical Programming*, 61:301–325, 1993.

[18] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, New York, 1988.

[19] R.T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, New Jersey, 1970.

[20] A. Ruszczyński. A regularized decomposition method for minimizing a sum of polyhedral functions. *Mathematical Programming*, 35:309–333, 1986.

[21] R. Schultz. Continuity properties of expectation functions in stochastic integer programming. *Mathematics of Operations Research*, 18:578–589, 1993.

[22] R. Schultz. On structure and stability in stochastic programs with random technology matrix and complete integer recourse. *Mathematical Programming*, to appear.

[23] D. Shannon and M. Sweedler. Using Gröbner bases to determine algebra membership, split surjective algebra homeomorphisms determine birational equivalence. *Journal of Symbolic Computation*, 6:267–273, 1988.

[24] B. Sturmfels. Gröbner Bases and Convex Polytopes. Lecture Notes, Holiday Symposium at New Mexico State University, Las Cruces, December 1994.

[25] S. Tayur. A new algorithm to solve stochastic integer programs with application to plant management. Technical report, Carnegie Mellon University, Pittsburgh, (in preparation).

[26] R. Thomas. A geometric Buchberger algorithm for integer programming. *Mathematics of Operations Research*, to appear.

[27] R. Urbaniak, R. Weismantel, and G. Ziegler. A variant of Buchberger's algorithm for integer programming. Preprint SC-94-29, ZIB, Berlin, 1994.

[28] S.W. Wallace and R.J-B. Wets. Preprocessing in stochastic programming: the case of linear programs. *ORSA Journal on Computing*, 4:45–59, 1992.