Mathematical Programming 69 (1995) 369–401

An algebraic geometry algorithm for scheduling in presence of setups and correlated demands

Sridhar R. Tayur [a,*], Rekha R. Thomas [b], N.R. Natraj [a]

[a] *Graduate School of Industrial Administration, Carnegie Mellon University, Schenley Park, Pittsburgh, PA 15213-3890, USA*

[b] *School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853, USA*

Received 9 December 1992; revised manuscript received 1 July 1994

Abstract

We study here a problem of scheduling n job types on m parallel machines, when setups are required and the demands for the products are correlated random variables. We model this problem as a chance constrained integer program.

Methods of solution currently available—in integer programming and stochastic programming—are not sufficient to solve this model exactly. We develop and introduce here a new approach, based on a geometric interpretation of some recent results in Gröbner basis theory, to provide a solution method applicable to a general class of chance constrained integer programming problems.

Our algorithm is conceptually simple and easy to implement. Starting from a (possibly) infeasible solution, we move from one lattice point to another in a monotone manner regularly querying a membership oracle for feasibility until the optimal solution is found. We illustrate this methodology by solving a problem based on a real system.

Keywords: Scheduling; Chance constrained programming; Integer programming; Decomposition; Gröbner basis; Polynomial ideals

1. Introduction

Firms are becoming increasingly aware that effective management of capacity constrained resources on the shop floor is essential to remain profitable in the presence of inherent variability of the marketplace. There are many instances, especially in the plants of industrial suppliers, where a set of jobs whose demands are random and

* Corresponding author.

correlated have to be scheduled on a bank of machines, not necessarily identical, with a minimum loss of capacity and minimum costs due to setups while meeting a high level of service. This is especially typical in the automotive parts supply industry as well as in the laminate industry (that supplies to Printed Circuit Board manufacturers), where a large group of suppliers compete on price and service.

The literature has recorded many models that consider the setup issues in presence of known demand. Several approaches dealing with these hard problems have resulted in implementable algorithms. In many cases, these results are applicable to situations with low variability of demand and low correlation between job types. However, these results are not transferable to a scenario where the job types that have low setup times or costs on a particular machine can have a strong positive correlation in demand because of their similarity in features. If these jobs are scheduled on the same machine based solely on setup cost information and mean demand values, the positive correlation in demand between products may result in poor service as the capacity of the machine on which they are scheduled constrains a quick response. Similarly, scheduling two job types with negative correlation of demand on the same machine is preferable from a demand management point of view; however, the setup costs and times involved may be high. While the variability of demand can be reduced by managing the customer ordering process appropriately (for example, by negotiating a contract for pre-committing a certain fixed amount every period), the correlation structure usually cannot be affected by supplier firms as the correlations arise due to inherent end-product features, and so this is a problem that cannot be resolved by negotiations with customers.

Consider the following scenario with three product types and two machines to illustrate the situation. Let products A, B and E with (random) demands D_A, D_B and D_E (in any period) have means $\hat{D}_A$, $\hat{D}_B$ and $\hat{D}_E$. Let machines α and β have capacities C_α and C_β respectively. Let the setup times for the products be S_{ij}, and the setup costs be K_{ij} for $i = A, B, E$ and $j = \alpha, \beta$. Let the processing times per unit be 1 on both machines for all product types. Suppose products A and B are scheduled on machine α and product E is scheduled on machine β. From a deterministic point of view, the cost in any period is $K_{A\alpha} + K_{B\alpha} + K_{E\beta}$. This schedule is an optimal solution to a deterministic problem of minimizing setup costs if, for example, $K_{A\alpha} < K_{A\beta}$, $K_{B\alpha} < K_{B\beta}$, $K_{E\alpha} > K_{E\beta}$ and $\hat{D}_A + \hat{D}_B \leqslant C_\alpha - S_{A\alpha} - S_{B\alpha}$, $\hat{D}_E \leqslant C_\beta - S_{E\beta}$. From a stochastic viewpoint, however, there is a possibility that demands will exceed the allotted production capacity in any period, resulting in a *shortfall*. When this happens, (1) the excess demand is backlogged (and so incur penalty costs), or (2) the excess demand is satisfied from inventory which in turn needs to be replenished (and so incur inventory costs), [1] or (3) the excess demand is lost (and so lose the margin and possibly market share) or (4) overtime is required to make up the shortfall (and so pay overtime costs). The point is, in all cases, there are costs incurred due to exceeding the capacity in any period, and

[1] In a dynamic infinite horizon capacitated model it can be shown that the amount of inventory (as safety stock) required depends on the shortfall distribution. Our model here should not be considered as static.

regardless of the specific remedial strategy the firm chooses, there is reason to schedule jobs on machines to avoid frequent occurrences of a shortfall. In the schedule discussed above, the shortfall probability is $(1 - \text{Probability}\{D_A + D_B \leqslant C_\alpha - S_{A\alpha} - S_{B\alpha}, D_E \leqslant C_\beta - S_{E\beta}\})$. This value of shortfall probability may be higher than the firm finds reasonable. A different schedule may satisfy the maximum allowable shortfall probability but may cost more in setups. The trade-off for the firm, therefore, is between shortfall management required due to demand correlation (and variance) and costs incurred due to setups.

Rather than prescribe an optimal value for γ, the probability of no shortfall, we will provide the optimal scheduling of jobs to machines for every γ. This is because a value of γ that a firm selects depends on the strategy and costs of making up (or losing) excess demand. It should be noted that the effects of correlation and variation in presence of capacity constraints and multiple items on performance characteristics can be often nonintuitive. In spite of this, many crucial capacity allocation and scheduling decisions on the shop floor are made without a sound scientific analysis. Often one relies on intuition or approximate extensions of deterministic models due to the lack of appropriate models and solution methods to tackle problems of this type. It is this void that we attempt to fill in this paper by developing a model and a solution methodology that are broadly applicable.

Our model formulation and the overall solution method is presented in detail in Section 2. Our formulation provides an increased flexibity in scheduling by allowing *lot splitting*. The total demand of a particular job type, if necessary, is split into pieces (or lots), and each lot is scheduled on a different machine. In an uncapacitated setting, there would be no incentive to split into lots, i.e., every job type would be made on a particular machine and the solution is straightforward (Section 2.3). However, the presence of capacity constraints and the desire to lower the frequency of shortfall occurrences make it beneficial to allocate capacity on several machines for a given product type.

The solution to our problem of scheduling in the presence of setups and correlated demands, therefore, is the set of optimal solutions to a sequence of chance constrained integer programs, each at a different level γ. We show that the set of optimal schedules for different γ are level sets; a particular schedule remains optimal for an interval of values of γ. Our solution method differs significantly from available techniques. Indeed, impressive advances in stochastic programming—in particular chance constrained programming with continuous variables and normal demands—have helped in solving many practical problems that have been formulated as convex programs with chance constraints [12]. However, while these methods have been successful without the integrality constraint, no satisfactory solution method exists for models that have integer variables and nonseparable chance (probabilistic) constraints arising from nonnormal distributions. In our setting, (1) the probabilistic constraint is not separable, (2) with only a handful of large customers ordering in batches, the normality assumption on demand may be violated significantly and (3) integer variables are required to model setup decisions. A subsequent section reviews some of the available techniques. The main technical contribution of this paper is to provide a novel method to solve this class of problems.

Our formulation is an instance of the following general problem, whose constraints can be separated into four categories:

$$\text{(P0)} \qquad \text{minimize} \quad \sum_{i=1}^{n} c_i x_i$$

$$\text{subject to:}$$

$$\text{(C1)} \quad \sum_{i=1}^{n} a_{ij} x_i \leqslant b_j, \quad j = 1, \ldots, m_1$$

$$\text{(C2)} \quad \sum_{i=1}^{n} a_{ij} x_i \leqslant b_j, \quad j = m_1 + 1, \ldots, m_1 + m_2$$

$$\text{(C3)} \quad (x_1, \ldots, x_n) \in \mathcal{S}$$

$$\text{(C4)} \quad x_i \in \mathbb{N}, \quad i = 1, \ldots, n,$$

where $\mathcal{S}$ is a set in $\mathbb{N}^n$, and whether or not any $(x_1, \ldots, x_n)$ belongs to $\mathcal{S}$ can be verified efficiently by a membership oracle. Note that $\mathcal{S}$ will depend on γ. Without constraint (C3), the problem is an integer program (IP). While constraints (C1) and (C2) look similar, the IP is easily solvable without the *complicating* constraints (C2). Furthermore, the structure of the constraints (C2) is similar to the constraints that define S. This structure lends itself to a *decomposition* that separates the constraints (C1) from (C2), and stores constraints (C2) by expanding the (available) membership oracle for (C3). We refer to the integer program obtained by removing constraints (C2) and (C3) from (P0) as the *reduced* IP. The solution method for (P0) first obtains an optimal solution of the reduced IP. If this is infeasible for (P0), we walk to other solutions of the reduced IP by moving along certain specified directions, querying the membership oracle each time to check feasibility with respect to (P0). We discuss this in detail shortly.

An essential component of our solution technique, therefore, is to find an appropriate set of directions which can be used to walk in a systematic manner from the optimal solution of the reduced IP to other solutions. These directions arise from a geometric interpretation of an algebraic algorithm to solve integer programs problems due to Conti and Traverso [2]. This algebraic algorithm uses the theory of Gröbner basis of polynomial ideals to solve integer programs. Starting at an integer point (infeasible for (P0)), the algorithm traces a path to the optimal solution of (P0) using only those directions specified by the associated Gröbner basis. In our case the integer program in question is the reduced IP whose optimal solution can be found quickly due to special structure of the problem. The elements of the Gröbner basis given by the Conti–Traverso algorithm can be interpreted geometrically as a set of directions [2] that can be used to trace paths

[2] More general results on Gröbner basis methods for integer programming appear in Thomas [11]. We use here only a few results relevant to our problem.

from every nonoptimal solution to an integer program to the optimal solution. Therefore, we can walk back from the optimal solution to every other feasible solution of the integer program by simply reversing these paths. Our algorithm terminates when all paths (starting from the optimal point of the reduced IP) are pruned. A path can be pruned in three ways: (1) it enters the feasible region of (P0) (satisfies the expanded membership oracle); (2) it lands on a point that violates the nonnegativity constraint; and (3) it lands on a point (still infeasible for (P0)) whose cost is higher than the best feasible solution found so far. A straightforward comparison of costs of points that are at the leaves of the paths that have entered the feasible region provides the optimum. If all the paths have been pruned and no feasible solution has been found, then the problem is infeasible.

To guarantee that this method always terminates at the optimum for (P0) or finds the problem infeasible (starting from the optimum for the reduced IP), we need to prove the following. *One*, reversing the directions and walking from the reduced IP optimum using these (reversed) directions always lands on a point that is feasible for the reduced IP if nonnegativity constraints are satisfied. *Two*, all feasible points of the problem can be reached using only these directions. *Three*, on every path, every step deeper into the polytope (starting from the reduced IP optimum) increases the cost monotonely. We show that all the above are indeed true.

The solution method outlined above is especially appropriate to the model we formulate for the following additional reasons. The Gröbner basis does not depend on the right-hand side of the IP constraints but only on the costs and the constraint matrix. In our formulation, the right-hand side can be varied based on management judgement on lot splitting. Furthermore, as we vary γ, we need to solve a nested set of integer programs with chance constraints, and the expanded membership oracle (which stores the probabilistic constraint as well as the complicating constraints) can be designed in such a way that all these nested problems are solved without solving for the Gröbner directions each time.

In the past, a general pessimism on the usefulness of Gröbner basis algorithms has arisen from the fact that the number of basis elements can be very large and the computational time to obtain these elements can be prohibitive. As will become clear in Section 3, we need a set of *starting generators* to compute the Gröbner basis. Indeed, a naive application of available methods to choose the starting generators makes the approach computationally infeasible. However, we show that our formulation has sufficient special structure that enables us to provide a set of starting generators that find the Gröbner basis elements quickly.

The rest of this paper is organized as follows. Section 2 presents the model, the overview of the solution procedure and a critique of available methods. In Section 3, we provide a geometric algorithm based on recent results in Gröbner basis theory, derive a good set of starting generators and provide an example to illustrate the overall procedure. In Section 4, we construct an efficient membership oracle and solve a problem based on a real system. Section 5 discusses another possible application of this methodology and concludes this paper.

2. The model

2.1. Formulation

Formally, the problem is to schedule n job types (indexed by i), whose demand in any given period is a random vector $(D_1, \ldots, D_n)$, on m machines (indexed by j) that have a capacity of C_j, $j = 1, \ldots, m$. The probability distribution of demand of the n job types in any period is $F(x_1, \ldots, x_n) = \text{Probability}\{D_1 \leqslant x_1, \ldots, D_n \leqslant x_n\}$, with means $(\hat{D}_1, \ldots, \hat{D}_n)$. The setup time for job type i on machine j is S_{ij}. Let K_{ij} denote the setup cost for job type i on machine j. To facilitate lot splitting, we have M_i, $i = 1, \ldots, n$, a set of management specified integers, that limits the maximum pieces (or lots) the demand of job type i can be partitioned into. Thus, if $M_1 = 4$, then on any machine only multiples of $\frac{1}{4}$ of the demand for product type 1 can be produced. Let L'_{ij} be the cost of producing a unit of product type i on machine j, and define $L_{ij} = (\hat{D}_i/M_i)L'_{ij}$. The processing time for a unit of job type i on machine j is p_{ij}. Let γ be the probability of no shortfall. The problem is modeled as the following chance constrained program:

$$\text{(P1)} \qquad \text{minimize} \quad \sum_i \sum_j \left(K_{ij} z_{ij} + L_{ij} y_{ij}\right)$$

subject to:

$$(1) \quad \sum_{j=1}^{m} y_{ij} = M_i, \quad \forall i$$

$$(2) \quad M_i z_{ij} \geqslant y_{ij}, \quad \forall i, j$$

$$(3) \quad \sum_{i=1}^{n} p_{ij}\left(\hat{D}_i/M_i\right) y_{ij} + \sum_{i=1}^{n} S_{ij} z_{ij} \leqslant C_j, \quad \forall j$$

$$(4) \quad \text{Prob}\left\{ \sum_{i=1}^{n} p_{ij}(D_i/M_i) y_{ij} + \sum_{i=1}^{n} S_{ij} z_{ij} \leqslant C_j, \forall j \right\} \geqslant \gamma,$$

$$(5) \quad z_{ij} \in \{0, 1\}, \quad y_{ij} \in \{0, 1, \ldots, M_i\}.$$

Variable z_{ij} equals 1 if job type i is scheduled on machine j; otherwise it is set to zero. Variable y_{ij}, an integer between zero and M_i, represents how many multiples of $1/M_i$ of demand of product i are scheduled on machine j. Thus, if in a period the demand realized for product i is D_i, $(y_{ij}/M_i)\ D_i$ will be scheduled on machine j. In our setting, therefore, allocation of capacity and scheduling go hand in hand.

The objective is to minimize the expected costs due to setups and production. Eq. (1) states that all of the demand for product i needs to be scheduled. Eq. (2) states that production for job type i cannot take place on machine j unless the appropriate setup has occurred. Eq. (3) states that the average demand allocated to a machine does not exceed its capacity. These are the complicating constraints of (P1). Eq. (4), the probabilistic constraint, states that the probability of production time and setup time not exceeding the capacities on the respective machines is at least γ. Notice the similarity in structure between constraint (3) and constraint (4).

In the framework of problem (P0), Eq. (1)–(2) constitute the set (C1), Eq. (3) is the set (C2), Eq. (4) corresponds to (C3) and Eq. (5) corresponds to (C4). Thus, the *reduced IP* consists of constraints (1), (2) and (5) only.

Note that doubling M_i provides more flexibility in scheduling due to more choices of lot-splitting. However, if M_i and M_i' are relatively prime with $M_i < M_i'$, it is possible that there is a feasible schedule with M_i but not with M_i'. To avoid computing the Gröbner basis several times when some M_i is varied (in a study of the effect of M_i on costs) we can define $M \geqslant \max_{i=1,\ldots,n} M_i$, over all possible candidates for M_i that we are interested in. In eq. (2), we can replace M_i by M. This assures that the constraint matrix of the reduced IP does not vary.

2.2. Overview of solution method

Our solution method separates constraints (1)–(2) and (5) from (3)–(4). The motivation for this decomposition is the fact that the reduced IP is trivial to solve, while adding constraint (3) to it makes the problem *NP-Complete* (recall the knapsack problem). The similarity in structure between (3) and (4) makes it especially attractive to move constraint (3) into the membership oracle. Samples from the demand distribution are appropriately stored to make up the membership oracle. In case we do not have an empirical distribution, we can obtain a desired accuracy for the probabilistic constraint at level γ by selecting an appropriate number of samples from $F(x_1, \ldots, x_n)$ [5]. It will be shown that a further decomposition of the reduced IP (by jobs) is possible. This will imply that the elements of the Gröbner basis for the reduced IP can be obtained by concatenation of n (smaller) Gröbner bases, one basis for each job type—a major computational advantage. This will be discussed in detail in Section 3.3.

Let $(y_{ij}^{\mathrm{r}}, z_{ij}^{\mathrm{r}})$ be the optimal solution to the reduced IP (superscript r corresponds to "reduced"). Starting from this point, if infeasible for (3)–(4), we use the Gröbner directions and walk on the lattice. At each step, we query the membership oracle. When all paths are pruned, we can obtain the required optimal solution by a straightforward comparison of active feasible points, or claim that the problem is infeasible.

2.3. The reduced IP

Recall that the reduced IP consists of constraints (1)–(2) and (5). This problem decomposes easily by product types. The optimal solution can be obtained by first observing that because of linear costs of production, a fixed charge for setup and no capacity constraint, there is no reason to split lots. Thus, for each job type, we simply find the machine on which this job type can be produced in the cheapest manner and call it β_i (in case there is a tie, we pick the lowest index). Thus (for each i), $(K_{i\beta_i} + M_i L_{i\beta_i}) \leqslant (K_{ij} + M_i L_{ij})$, $\forall j$, and β_i is the smallest j for which this is true. The optimal solution to the reduced IP is obtained by setting $z_{i\beta_i}^{\mathrm{r}} = 1$, $y_{i\beta_i}^{\mathrm{r}} = M_i$ and the rest of the variables to zero. If this satisfies the complicating constraints (Eq. (3)) as well as the probabilistic constraint (Eq. (4)), then it is the optimal solution to (P1).

2.4. Review of existing solution methods

As stated in the Introduction, the results of deterministic scheduling models do not carry over to our situation. More importantly, the techniques commonly used in stochastic programming also fail in our setting. Recall that our probabilistic constraint is of the form $P\{Hx \leqslant h\} \geqslant \gamma$, with a random matrix H and integer variables x. We briefly review four available methods for solving chance constrained programming problems and point out their inapplicability here. See [12] for details and references for the descriptions below.

Consider first the approach, useful for problems with continuous variables, that constructs a deterministic equivalent. This converts the problem into a convex nonlinear programming problem which can be solved by gradient methods. Our problem may not have a deterministic equivalent of the type usually studied by stochastic programmers. This is because we have a joint constraint, rather than a constraint for each $j = 1, \ldots, m$ and the demands typically arise from a distribution other than the multivariate normal.

A second method, a dual approach to solve continuous variable problems in presence of joint chance constraints, requires the coefficient matrix in the chance constraint to be deterministic. In our formulation, however, the coefficient matrix in the probabilistic constraint is random. It has been shown that when this coefficient matrix is random, the set of feasible solutions that satisfy the probabilistic constraint is nonconvex in general, thus precluding the use of convex optimization techniques.

A third approach, based on outer approximations for mixed integer nonlinear problems has been successfully applied for problems arising in chemical engineering. This method, however, requires a deterministic equivalent, convexity of the feasible region, as well as an explicit function that can be differentiated. These conditions are not satisfied by the problem at hand. See [4] for details of this method.

A fourth approach, based on recent results on rapidly mixing Markov chains, finds a near optimal solution for a certain class of chance constrained programs (with continuous variables) using a randomized algorithm. See [6] for details.

To summarize, although many available methods can be adapted to heuristically solve complex integer nonlinear problems that may arise from chance constrained models, no particular one solves our problem exactly. Ultimately, we believe that all these approaches should form part of a tool kit for analyzing complex problems modeled in this fashion.

3. Solution method

3.1. The walk back procedure

In this section we describe a systematic procedure to walk from the optimum of an integer program to any other feasible lattice point in its feasible region. Given a feasible lattice point α, the path to this point from the optimum is obtained by moving from one feasible solution to another according to certain rules. The objective function value of successive lattice points in this path increases monotonically. In Section 2.3 we showed that the optimal solution of the reduced IP was easy to find but that it may be infeasible with respect to the constraints in the expanded membership oracle. Since every feasible solution of the overall problem (P1) is also feasible for the reduced IP, we can use the above-mentioned paths to walk from the optimum of the reduced IP, to feasible solutions of (P1). We give an algorithm that uses these paths to find a sufficient set of feasible solutions to (P1), from which the optimal solution can be identified easily.

Consider an integer program of the form:

$$\begin{array}{ll} \min & cx \\ \text{s.t.} & Ax = b \\ & x \in \mathbb{N}^n, \end{array}$$

where $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$ and $c \in \mathbb{R}^n$. Let IP denote all integer programs of the above form, obtained by varying the right-hand side vector b but keeping A and c fixed. Let IP(b) denote that integer program from the above family with right-hand side vector b. We assume that each IP(b) is bounded with respect to the cost function c.

Consider the map $\pi : \mathbb{N}^n \mapsto \mathbb{Z}^m$ such that $\pi(x) = Ax$. Given a vector $b \in \mathbb{Z}^m$, the set of feasible solutions to IP(b) constitute $\pi^{-1}(b)$, the pre-image of b under this map. We identify $\pi^{-1}(b)$ with its convex hull and call it the *b-fiber* of IP. Therefore the b-fiber of IP is just the convex hull of all feasible solutions to IP(b).

Definition 1. We call $>$ a *term order* on $\mathbb{N}^n$ if $>$ has the following properties:
(1) $>$ is a total order on $\mathbb{N}^n$;
(2) $\alpha > \beta \Rightarrow \alpha + \omega > \beta + \omega$ for all $\alpha, \beta, \omega \in \mathbb{N}^n$ (i.e., $>$ is compatible with sums);
(3) $\alpha > 0$ for all $\alpha \in \mathbb{N}^n \setminus \{0\}$ (i.e., 0 is the minimum element).

Each point in $\mathbb{N}^n$ is a feasible solution in some fiber of IP—i.e., α in $\mathbb{N}^n$ is a feasible solution in the $A\alpha$-fiber of IP. Suppose we now group points in $\mathbb{N}^n$ according to increasing cost value cx. Since more than one lattice point can have the same cost value, this ordering is not a total order on $\mathbb{N}^n$. We can, however, refine this to create a total order by adopting some term order $>$, to break ties among points with the same cost value. Let $>_c$ denote this composite total order on $\mathbb{N}^n$ that first compares two points by the cost c and breaks ties according to $>$. We may think of $>_c$ as a refinement of the cost function c. It can be checked that $>_c$ satisfies properties (1) and (2) of Definition 1. However, $>_c$ may not satisfy (3) and therefore, $>_c$ is not always a term order on $\mathbb{N}^n$.

Example. Consider $\mathbb{N}^2$ and $c = (-1, 2)$.

$$\left.\begin{array}{l} c.(0,0) = 0 \\ c.(1,0) = -1 \end{array}\right\} \Rightarrow (0,0) >_c (1,0).$$

We now replace the objective function of programs in IP by its refinement $>_c$. This change does not affect the optimal value of a member of IP, but since $>_c$ is a total order on $\mathbb{N}^n$ it insures a unique optimum in every fiber of IP. Let $\mathcal{S}$ denote the set of all points in $\mathbb{N}^n$ that are nonoptimal with respect to $>_c$ in the various fibers of IP. The following lemma characterizes the structure of $\mathcal{S}$.

Lemma 1. *There exists a unique, minimal, finite set of vectors $\alpha(1),\dots,\alpha(s) \in \mathbb{N}^n$ such that the set of all nonoptimal points with respect to $>_c$ in all fibers of IP is a subset of $\mathbb{N}^n$ of the form*

$$\mathcal{S} = \bigcup_{i=1}^{s} (\alpha(i) + \mathbb{N}^n).$$

Proof. First we show that if $\alpha \in \mathcal{S}$, then $\alpha + \mathbb{N}^n \subset \mathcal{S}$. This will imply that $\mathcal{S}$ is an order ideal. If $\alpha \in \mathcal{S}$, then α is a nonoptimal point with respect to $>_c$ in $\pi^{-1}(A\alpha)$. Let β be the unique optimum in this fiber with respect to $>_c$. Consider $\alpha + \omega$, $\beta + \omega$ for $\omega \in \mathbb{N}^n$. Then

(1) $A(\alpha + \omega) = A(\beta + \omega)$ since $A\alpha = A\beta$.
(2) $(\alpha + \omega), (\beta + \omega) \in \mathbb{N}^n$ since $\alpha, \beta, \omega \in \mathbb{N}^n$.
(3) $\alpha >_c \beta \Rightarrow \alpha + \omega >_c \beta + \omega$.

(1)–(3) implies that $\alpha + \omega$ is not the optimal lattice point in the fiber $\pi^{-1}(A(\alpha + \omega))$. Therefore $\alpha + \omega \in \mathcal{S}$. Since ω is an arbitrary element in $\mathbb{N}^n$, this establishes that $\alpha + \mathbb{N}^n \subset \mathcal{S}$. By Dickson's Lemma, every order ideal in $\mathbb{N}^n$ has finitely many minimal elements. Applying this to $\mathcal{S}$, we conclude that there exists a unique minimal set of points $\alpha(1),\dots,\alpha(s) \in \mathcal{S}$ such that

$$\mathcal{S} = \bigcup_{i=1}^{s} (\alpha(i) + \mathbb{N}^n). \quad \square$$

We now define a *test set* for IP dependent on the matrix A and a refined objective function $>_*$. [3]

Definition 2. A set $\mathcal{G}_{>_*} \subseteq \mathbb{Z}^n$ is a *test set* for the family of integer programs IP (with respect to the matrix A and refined cost function $>_*$) if

(I) for each nonoptimal point α in each fiber of IP, there exists $g \in \mathcal{G}_{>_*}$ such that $\alpha - g$ is a feasible solution in this fiber and $\alpha >_* \alpha - g$ and
(II) for the optimal point β in a fiber of IP, $\beta - g$ is infeasible for every $g \in \mathcal{G}_{>_*}$.

A test set for IP gives an obvious algorithm to solve an integer program in IP provided we know a feasible solution to this program. At every step of this algorithm, there either exists an element in the test set which when subtracted from the current solution yields an improved solution, or there will not exist such an element in the set.

[3] See [7] for a different test set.

In the former case the solution at hand is nonoptimal whereas in the latter case it is optimal. If a new solution is obtained, we repeat until the process stops at the unique optimum of the fiber. Since each integer program considered is bounded with respect to the objective function $>_c$, the procedure is guaranteed to be finite.

We construct a set $\mathcal{G}_{>_c}$ for IP as follows. Let $\mathcal{G}_{>_c} = \{g_i = (\alpha(i) - \beta(i)),\ i = 1, \ldots, s\}$, where $\alpha(1), \ldots, \alpha(s)$ are the unique minimal elements of $\mathcal{S}$ and $\beta(i)$ is the unique optimum with respect to $>_c$ in the $A\alpha(i)$-fiber of IP. We show below that $\mathcal{G}_{>_c}$ is indeed a test set for IP and in the next subsection we prescribe an algorithm that computes $\mathcal{G}_{>_c}$ without explicitly determining $\alpha(i)$ and $\beta(i)$. Notice that $(\alpha(i) - \beta(i))$ is a lattice point in the subspace $\mathcal{S} = \{x \in \mathbb{Z}^n\colon\ Ax = 0\}$. Geometrically we can think of $(\alpha(i) - \beta(i))$ as the directed line segment $\vec{g_i} = \overrightarrow{[\alpha(i),\ \beta(i)]}$ in the $A\alpha(i)$-fiber of IP. The vector is directed from the nonoptimal point $\alpha(i)$ to the optimal point $\beta(i)$ due to the minimization requirement in the problem.

Observation 1. Adding a vector $v \in \mathbb{N}^n$ to both end points of $\vec{g_i}$ translates $\vec{g_i}$ from the $A\alpha(i)$-fiber of IP to the $A(\alpha(i) + v)$-fiber of IP.

We now consider an arbitrary fiber of IP and a feasible lattice point δ in this fiber. For each vector $\vec{g_i}$ in $\mathcal{G}_{>_c}$, check if it can be translated through some vector v in $\mathbb{N}^n$ such that the tail of the translated vector is incident at δ. At δ draw all such possible translations of vectors from $\mathcal{G}_{>_c}$. Notice that the head of a translated vector is also incident at a feasible lattice point in the same fiber as δ. We do this construction for all feasible lattice points in all fibers of IP. From Lemma 1 and the definition of $\mathcal{G}_{>_c}$, it follows that no vector in $\mathcal{G}_{>_c}$ can be translated by $v \in \mathbb{N}^n$ such that its tail meets the optimum on a fiber. We now state the main result of this section.

Theorem 1. *The above construction builds a connected, directed graph in every fiber of IP. The nodes of the graph are all the lattice points in the fiber and the edges are the translations of elements of* $\mathcal{G}_{>_c}$ *by nonnegative integral vectors. The graph in a fiber has a unique sink at the opimum in the fiber with respect to* $>_c$ *and the outdegree of every nonoptimal lattice point in the fiber is at least one.*

Proof. Pick a fiber of IP and at each feasible lattice point construct all possible translations of the vectors $\vec{g_i}$ from the set $\mathcal{G}_{>_c}$ as described earlier. This results in a possibly disconnected directed graph in this fiber where nodes are the lattice points and edges the translations of elements in $\mathcal{G}_{>_c}$. Let α be a nonoptimal lattice point in this fiber. By Lemma 1, $\alpha = \alpha(i) + v$ for some $i \in \{1, \ldots, s\}$ and $v \in \mathbb{N}^n$. Now $\alpha' = \beta(i) + v$ also lies in this fiber and $\alpha >_c \alpha'$ since $\alpha(i) >_c \beta(i)$. Therefore $\vec{g_i}$ translated by $v \in \mathbb{N}^n$ is an edge in this graph and we can move along it from α to an improved point α' in the same fiber. This proves that from every nonoptimal lattice point in the fiber we can reach an improved point in the same fiber by moving along an edge of the graph. The structure of $\mathcal{S}$ and the construction of the set $\mathcal{G}_{>_c}$ imply that the outdegree of the optimal solution is zero. Therefore, a directed path from a nonoptimal point terminates precisely at the unique optimum of the fiber. This in turn implies connectedness of the graph. □

We call the graph in the b-fiber of IP the $>_c$*-skeleton* of that fiber to denote the dependence of the graph on the refined objective function $>_c$. The following corollaries follow from Theorem 1.

Corollary 1. *In the $>_c$-skeleton of a fiber, there exists a directed path from every nonoptimal point α to the unique optimum β. The objective function value (with respect to c) of successive points in the path decreases monotonically from α to β.*

Corollary 2. *The set $\mathscr{G}_{>_c}$ is a uniquely defined minimal test set for IP.*

Example 1. Consider the integer program:

$$\begin{array}{ll} \text{minimize} & 100a + 0b + 0c + 100d \\ \text{subject to} & \\ & 2a + b + c + 3d = 30 \\ & a + c + 5d = 16 \\ & a,\ b,\ c,\ d \text{ nonnegative and integer.} \end{array}$$

The feasible region of the integer program are the integer points in the (30, 16)-fiber. The extreme points are given by (a, b, c, d): (14,0, 2, 0), (0, 14, 16, 0), (0, 20, 1, 3), (1, 19, 0, 3) and (11, 5, 0, 1).

To illustrate graphically, we draw the fiber by projecting onto the b–d co-ordinate plane (Fig. 1). The extreme points on this plane are: (0, 0), (14, 0), (20, 3), (19, 3) and (5, 1). All lattice points in the projection lift to become lattice points in the actual fiber and vice-versa.

The test directions for this example are $\overrightarrow{[(0,2,0,1),(0,0,5,0)]}$ (direction P) and $\overrightarrow{[(1,0,0,0),(0,1,1,0)]}$ (direction Q). The (projections of) test directions in the b–d plane are $\overrightarrow{[(2,1),(0,0)]}$ and $\overrightarrow{[(1,0),(1,0)]}$ and (the projection of) the point (0, 14, 16, 0), for example, is (14, 0). Fig. 1 also shows some paths (not the entire

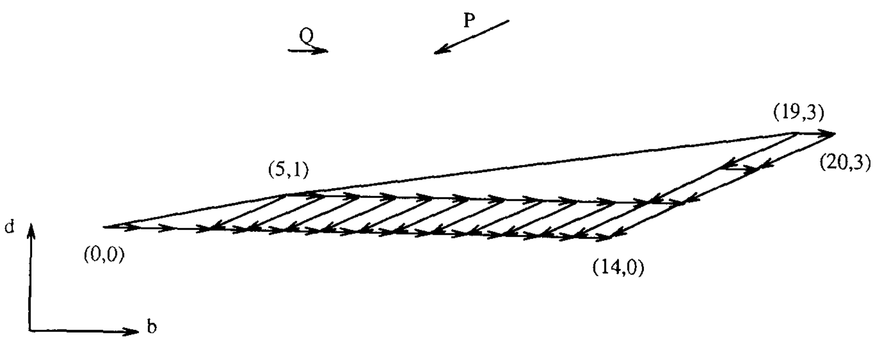

Fig. 1. Projection on $b-d$ plane for Example 1.

skeleton) from some feasible solutions (including all the extreme points) to the optimal solution using directions P and Q in the (30, 16)-fiber.

From point (0, 14, 16, 0), we cannot travel to any other feasible point in the (30, 16)-fiber using directions P and Q; this is because direction P leads to (0, 12, 21, −1) and direction Q leads to (−1, 13, 15, 0). This implies that (0, 14, 16, 0) is the optimal solution.

Consider the other extreme points. From (0, 20, 1, 3) direction P leads to (0, 18, 6, 2) and then to (0, 16, 11, 1) and finally to (0, 14, 16, 0). Direction Q, at all points in the path, leads to infeasible solutions (that violate nonnegativity) and is therefore not used.

From (11, 5, 0, 1), a repeated movement in direction Q (11 times) takes you to (0, 16, 11, 1). Now, direction P leads to (0, 14, 16, 0), the optimal solution.

From (1, 19, 0, 3), direction Q moves you to (0, 20, 1, 3). Now, repeated application of direction P (3 times) leads you to (0, 14, 16, 0).

Finally, from (14, 0, 2, 0), direction Q (14 times) leads to (0, 14, 16, 0).

Note that from any nonoptimal point, there may be several paths to the optimal solution in the (30, 16)-fiber.

We are now ready to propose a solution method for problem (P1) from Section 2.1. Recall from Section 2.3 the reduced IP and its optimal solution. The feasible region of the reduced IP is given by constraints (1)–(2) and (5) of (P1). Consider the $>_c$-skeleton of the appropriate fiber. Suppose we now reverse the direction of all edges in this skeleton and call the resulting graph the *reverse* $>_c$*-skeleton* of the fiber. Corollary 1 then implies that there exists a directed path $P(\alpha)$ in the reverse $>_c$-skeleton of this fiber, from the optimum of the reduced IP to the feasible lattice point α in the fiber. Also, the objective function value of points reached as the path is traversed from β to α, increases monotonically.

We first show that we can build the reverse $>_c$-skeleton in a fiber by reversing the orientations of vectors in the set $\mathscr{G}_{>_c}$ and repeating the construction described before Theorem 1. Let $\mathscr{G}'_{>_c}$ denote the set of vectors in $\mathscr{G}_{>_c}$ with directions reversed. At a feasible lattice point δ in a fiber of IP, we draw all vectors from $\mathscr{G}'_{>_c}$ that can be translated through some $v \in \mathbb{N}^n$ such that its tail is incident at δ. As before, the heads of the translated vectors are also incident at feasible lattice points in the same fiber. Since only the orientation of the vectors in the test set was reversed, this construction and the previous one have the same underlying undirected graph in every fiber. This proves the claim.

We now use these paths in the reverse $>_c$-skeleton of the reduced IP to find a sufficient set of feasible solutions, $\mathscr{Y}$, to (P1) from which the optimal solution to (P1) is obtained. Let β be the optimum of the reduced IP and $P(\alpha)$ be a path in the reverse $>_c$-skeleton of the fiber from β to α. Let $\mathscr{P}$ be a set of paths of the form $P(\alpha)$. We may assume that β is infeasible for (P1) since otherwise we are done.

```
**Algorithm:** Finding a set of feasible solutions to (P1)
**initialize**: (i) $\mathscr{P} = \{P(\beta)\}$,   (ii) $\mathscr{Y} = \emptyset$.
**repeat**
```

```
    For $P(\alpha) \in \mathscr{P}$, let $P(\omega_1), \ldots, P(\omega_q)$ be the paths obtained by adding to $P(\alpha)$ those
    edges in $\mathscr{S}'_{>c}$ such that $\omega_i \geqslant 0$. This insures that $P(\omega_i)$ is a path in the reverse $>_c$-
    skeleton. A path that leads to an infeasible point may be assumed to be pruned at $\alpha$.
      for $i$ from 1 to $q$
        (i) if $\omega_i$ feasible for (P1) let $\mathscr{Y} = \mathscr{Y} \cup \{\omega_i\}$ and prune $P(\omega_i)$.
              (feasibility is checked by querying the membership oracle)
        (ii) else if $\omega_i >_c y$ for some $y \in \mathscr{Y}$ then prune $P(\omega_i)$
              (the path if continued will give points that are more expensive than
              those in $\mathscr{Y}$)
        (iii) else $\mathscr{P} = \{\mathscr{P} \cup P(\omega_i)\}$.
    Delete $P(\alpha)$ from the set $\mathscr{P}$.
until all paths in $\mathscr{P}$ are pruned.
```

Clearly all points in $\mathscr{Y}$ are feasible for (P1). Finiteness of the above procedure follows from the fact that the optimal solution to (P1) is bounded with respect to $>_c$. Thus (P1) is infeasible if and only if $\mathscr{Y}$ is empty when the algorithm terminates. We now prove that the set $\mathscr{Y}$ contains an optimal solution to (P1).

Theorem 2. *The points in $\mathscr{Y}$ with the least cost value $c(y)$ are optimal for* (P1).

Proof. Let μ be any feasible solution to (P1) and let $y^* \in \mathscr{Y}$ be a point with least cost $c(y)$. Clearly μ is feasible for the reduced IP and there exists a path $P(\mu)$, in the reverse $>_c$-skeleton, from the optimum β of the reduced IP, to μ. Note that $P(\mu)$ may not be unique but any such path will suffice for this proof. We identify $P(\mu)$ with the sequence of nodes in the reverse $>_c$-skeleton that constitute it, i.e., $P(\mu) = \{\beta = u_1, \ldots, u_m = \mu\}$. Now let $j \in \{1, \ldots, m\}$ be the largest number such that $P(u_j)$ is in the set $\mathscr{P}$ of the above procedure at some stage. We have two cases to consider.

(i) $P(u_j)$ was pruned since u_j became feasible for (P1): In this case, $u_j \in \mathscr{Y}$ and $c(u_j) \geqslant c(y^*)$. Also $c(\mu) \geqslant c(u_j)$, since the successive nodes in $P(\mu)$ have monotonically increasing cost when the path is traversed from β to μ. Therefore $c(\mu) \geqslant c(y^*)$.

(ii) $P(u_j)$ was pruned since $u_j >_c y$ for some $y \in \mathscr{Y}$: Again, $c(\mu) \geqslant c(u_j) \geqslant c(y^*)$. Therefore, we have shown that in either case $c(\mu) \geqslant c(y^*)$ which proves the theorem. □

Using Theorem 2 we can identify the optimal solution to (P1) by a straightforward comparison of the costs of points in $\mathscr{Y}$.

We wish to emphasize that the solution method described for (P1) can be adapted in general for problems modeled in this way. Given an integer program in which some of the constraints are complicating and possibly nonlinear, it is possible to create a reduced IP and a membership oracle and use the above solution method. The efficiency of the procedure will depend heavily on how one separates the constraints into those that constitute the reduced IP and those that contribute to the membership oracle.

3.2. Computing test sets

In this subsection we prescribe an algorithm to compute the test sets described in Section 3.1 by establishing the connections of these test sets with certain algebraic concepts. An important concept in the theory of polynomial ideals is that of a *Gröbner basis* of an ideal [1] with respect to a *term order*. These special bases can be computed using *Buchberger's algorithm* [1] which has been implemented in most computer algebra packages. We show below that the test set $\mathcal{G}_{>_c}$ is equivalent to the *reduced Gröbner basis* of a certain polynomial ideal with respect to the composite order $>_c$. This immediately allows us to compute these test sets in practice, making the solution method proposed earlier, a feasible one.

In order to establish the equivalence, we first recall some basic definitions and concepts [3] from the theory of polynomial ideals. Let k be any field and let $k[x_1,\ldots,x_n]$ be the polynomial ring in n variables over k.

Definition 3. A subset $I \subset k[x_1,\ldots,x_n]$ is an *ideal* if it satisfies:
(i) $0 \in I$;
(ii) If $f, g \in I$, then $f+g \in I$;
(iii) If $f \in I$ and $h \in k[x_1,\ldots,x_n]$, then $hf \in I$.

Definition 4. Let $f_1,\ldots,f_s$ be polynomials in $k[x_1,\ldots,x_n]$. We say $f_1,\ldots,f_s$ is a *basis* for I if $I = \langle f_1,\ldots,f_s\rangle = \{\sum_{i=1}^{s} h_i f_i \colon h_1,\ldots,h_s \in k[x_1,\ldots,x_n]\}$.

For ease of exposition, we let the monomial $x_1^{\alpha_1}\ldots x_n^{\alpha_n}$ be denoted by x^α, where $\alpha \in \mathbb{N}^n$. We may identify a monomial in $k[x_1,\ldots,x_n]$ with its exponent vector $\alpha \in \mathbb{N}^n$. Recall from Section 3.1 the definition of a term order on $\mathbb{N}^n$. Under the above identification, $>$ can be thought of as a term order on the monomials of $k[x_1,\ldots,x_n]$; that is, $x^\alpha > x^\beta$ if and only if $\alpha > \beta$. This term order, therefore, identifies in every polynomial $f \in I$, a unique *initial term* denoted $in_>(f)$, which is the term of f involving the most expensive monomial with respect to $>$. We can now define the initial ideal of I with respect to $>$.

Definition 5. We call $init_>(I) = \langle in_>(f) \colon f \in I\rangle$, the *initial ideal* of I with respect to $>$.

Notice that $init_>(I)$ is generated by monomials.

Definition 6. We say $\mathcal{G}_> = \{g_1,\ldots,g_s\} \subset I$ is a *Gröbner basis* of I with respect to $>$ if $init_>(I) = \langle in_>(g_1),\ldots,in_>(g_s)\rangle$.

In other words, a Gröbner basis of I is a special basis of I with the property that $init_>(I)$ can be generated by the initial terms of the elements $g_1,\ldots,g_s$ in I. The condition on the initial terms of elements in $\mathcal{G}_>$ does imply that $\mathcal{G}_>$ is a basis of I. A Gröbner basis of an ideal is not necessarily unique. We can however talk of the *reduced Gröbner basis of I* with respect to $>$ which is unique.

Definition 7. The *reduced Gröbner basis* of I with respect to $>$ is a Gröbner basis $\mathscr{G}_>$ of I such that:

(i) $in_>(g_i)$ has unit coefficient for each $g_i \in \mathscr{G}_>$,

(ii) For each $g_i \in \mathscr{G}_>$, no monomial of g_i lies in $\langle init_>(\mathscr{G}_> - g_i)\rangle$.

Given any basis of an ideal and a term order $>$, Buchberger's algorithm computes the reduced Gröbner basis of the ideal with respect to this term order. We call the basis of the ideal needed as input to this algorithm, a *starting basis*.

Conti and Traverso [2] describe an algebraic algorithm to solve integer programs using the theory of Gröbner bases. We resort to their algorithm to obtain the ideal whose reduced Gröbner basis corresponds to the test set $\mathscr{G}_{>_c}$. Recall from Section 3.1 the family of integer programs IP with matrix $A \in \mathbb{Z}^{m \times n}$ and cost function c. We denote the jth column of A by a^j.

Consider the map

$$\varphi : k[x] = k[x_1, \ldots, x_n] \mapsto k\left[y^{a^1}, \ldots, y^{a^n}\right]$$

$$x_j \mapsto y^{a^j},$$

where $y = (y_1, \ldots, y_m)$ and $x = (x_1, \ldots, x_n)$. Let $J = kernel(\varphi) = \{f \in k[x]\colon \varphi(f) = 0\} \subseteq k[x]$. It may be checked that J is an ideal of $k[x]$. A binomial in $k[x]$ is the difference of two monomials. We now recall a few facts about the ideal J.

Observation 2. Let α, β be vectors in $\mathbb{N}^n$. Then, the binomial $x^\alpha - x^\beta$ is in J if and only if $A\alpha = A\beta$ where $\alpha, \beta \in \mathbb{N}^n$.

Given an integral vector $\alpha \in \mathbb{Z}^n$, we can write it uniquely as $\alpha = \alpha^+ - \alpha^-$ where α^+, $\alpha^- \in \mathbb{N}^n$ and have disjoint supports.

Lemma 2 (Sturmfels [10]). *The ideal J is generated by binomials of the form* $\{x^{\alpha^+} - x^{\alpha^-}\colon A\alpha = 0,\ \alpha \in \mathbb{Z}^n\}$.

Recall the test set $\mathscr{G}_{>_c} = \{(\alpha(i) - \beta(i)),\ i = 1, \ldots, s\}$, with respect to the composite order $>_c$, which sorted points by cost c and then broke ties using a fixed term order $>$. As in the identification of a monomial with its exponent vector, we identify the vector $(\alpha(i) - \beta(i))$ with the binomial $x^{\alpha(i)} - x^{\beta(i)}$. For each g_i in $\mathscr{G}_{>_c}$, both $\alpha(i)$ and $\beta(i)$ were feasible solutions from the same fiber of IP. This implies that $A\alpha(i) = A\beta(i)$, $\alpha(i), \beta(i) \in \mathbb{N}^n$ for all $i = 1, \ldots, s$. Therefore by Observation 2, $\langle x^{\alpha(i)} - x^{\beta(i)}$, $i = 1, \ldots, s\rangle \subseteq J$. Further, since $\alpha(i) >_c \beta(i)$ for all $i = 1, \ldots, s$, we have that $in_{>_c}(x^{\alpha(i)} - x^{\beta(i)}) = x^{\alpha(i)}$ for all i. Therefore, $\langle x^{\alpha(i)},\ i = 1, \ldots, s\rangle \subseteq init_{>_c}(J)$. We now show that in fact we have equality here.

Theorem 3. *The test set* $\mathscr{G}_{>_c} = \{x^{\alpha(i)} - x^{\beta(i)},\ i = 1,\dots,s\}$ *is the reduced Gröbner basis of* J *with respect to* $>_c$.

Proof. We show that $in_{>_c}(f)$ is in the monomial ideal $\langle x^{\alpha(i)},\ i = 1,\dots,s\rangle$ for each polynomial f in J. By Observation 2 and Lemma 2, $J = \langle x^\alpha - x^\beta\colon\ A\alpha = A\beta,\ \alpha,\beta \in \mathbb{N}^n\rangle$. We may assume that $in_>(x^\alpha - x^\beta) = x^\alpha$. In fact the binomials of the above form form a universal Gröbner basis for J and so it is enough to show that $x^\alpha \in \langle x^{\alpha(i)},\ i = 1,\dots,s\rangle$ for each binomial $x^\alpha - x^\beta \in J$. Now $in_>(x^\alpha - x^\beta) = x^\alpha$ implies that $\alpha >_c \beta$. Therefore α is a nonoptimal point with respect to $>_c$ in the $A\alpha$-fiber of IP. Therefore α is in $\mathscr{S}$, the set of all nonoptimal points from all fibers of IP. By Lemma 1, $\mathscr{S} = \bigcup_{i=1}^{s}(\alpha(i) + \mathbb{N}^n)$. This implies that $\alpha = \alpha(i) + v$ for some $i \in \{1,\dots,s\}$ and $v \in \mathbb{N}^n$. Therefore $x^{\alpha(i)}$ divides x^α which in turn implies that $x^\alpha \in \langle x^{\alpha(i)},\ i = 1,\dots,s\rangle$. Therefore $\mathscr{G}_{>_c}$ is a Gröbner basis for I with respect to $>_c$.

We now show that $\mathscr{G}_{>_c}$ is in fact the reduced Gröbner basis of I with respect to $>_c$. For $x^{\alpha(i)} - x^{\beta(i)} \in \mathscr{G}_{>_c}$, we first note that $x^{\alpha(i)} \notin \langle init_{>_c}(\mathscr{G}_{>_c} - g_i)\rangle$ since $\alpha(1),\dots,\alpha(s)$ were a unique minimal set of generators for $\mathscr{S}$. Also $x^{\beta(i)} \notin \langle init_{>_c}(\mathscr{G}_{>_c} - g_i)\rangle$ since $\beta(i)$ was the unique optimum with respect to $>_c$ on its fiber and so is not in $\mathscr{S}$. Therefore $\mathscr{G}_{>_c}$ is the reduced Gröbner basis for I with respect to $>_c$. □

In Section 3.1 we mentioned that $>_c$ is not always a term order on $\mathbb{N}^n$. The Buchberger algorithm requires a term order as input for the calculation of Gröbner bases to insure finite termination of the algorithm. Since $>_c$ may not always have 0 as the minimum element, there is a danger of finding an infinite descending chain of monomials with respect to $>_c$ in the ideal J. In our case this is prevented by the fact that programs in IP are bounded with respect to cost and that the tie breaking term order is in fact a legitimate term order. Therefore $>_c$ can be used instead of a term order as the order in the Buchberger algorithm. In Section 4, we use the computer algebra package *Macaulay* [9] to compute Gröbner bases.

3.3. A starting basis

In the previous subsection we showed that the test set $\mathscr{G}_{>_c}$ is equivalent, under certain identifications, to the reduced Gröbner basis of the ideal J with respect to $>_c$. As mentioned before, we can use the Buchberger algorithm to compute this reduced Gröbner basis of J. The Buchberger algorithm needs as input the order $>_c$ as well as a basis of J, which we call the starting basis. The order $>_c$ can be input to Macaulay by simply specifying the cost function c. Macaulay uses a default term order to break ties creating the composite order we require. A standard way to implicitly specify a basis of the ideal J associated with any integer program is given by the Conti–Traverso algorithm. This general approach was however seen to be rather infeasible from a computational point of view (see Section 4), except for small examples from the family of problems associated with the reduced IP from Section 2.3. We describe below the special starting basis we used to speed up calculations for the family of problems at hand.

Recall the reduced IP from Section 2.3.

$$\text{(I)} \qquad \text{minimize} \quad \sum_i \sum_j (K_{ij} z_{ij} + L_{ij} y_{ij})$$

subject to:

$$\sum_{j=1}^{m} y_{ij} = M_i, \quad i = 1, \ldots, n$$

$$M_i z_{ij} \geqslant y_{ij}, \quad i = 1, \ldots, n, \ j = 1, \ldots, m$$

$$z_{ij} \in \{0, 1\}, \quad y_{ij} \in \{0, 1, \ldots, M_i\} \text{ for all } i, j.$$

We convert this problem into the form of the general integer program in Section 3.1 by adding slack variables and forcing the requirement that the variables z_{ij} have a value zero or one. The problem can be restated as:

$$\text{(II)} \qquad \text{minimize} \quad \sum_i \sum_j (K_{ij} z_{ij} + L_{ij} y_{ij})$$

such that:

$$(1) \quad \sum_{j=1}^{m} y_{ij} = M_i, \quad i = 1, \ldots, n$$

$$(2) \quad y_{ij} + a_{ij} - M_i z_{ij} = 0, \quad i = 1, \ldots, n, \ j = 1, \ldots, m$$

$$(3) \quad z_{ij} + b_{ij} = 1, \quad i = 1, \ldots, n, \ j = 1, \ldots, m$$

$$y_{ij}, a_{ij}, z_{ij}, b_{ij} \in \mathbb{N}, \quad i = 1, \ldots, n, \ j = 1, \ldots, m.$$

The matrix A, associated with this integer program can be divided into four blocks according to the four sets of variables y_{ij}, a_{ij}, z_{ij} and b_{ij}. Let I denote the identity matrix of size mn and 0 denote a matrix with all entries zero. Let D be the block diagonal (0, 1)-coefficient matrix associated with the constraints (1) of system (II). Let $-MI$ denote the coefficient matrix of the variables z_{ij}, $i = 1, \ldots, n$ and $j = 1, \ldots, m$ in constraints (2) of system (II). We can then write A as the $(n + 2mn) \times (4mn)$ integer matrix:

$$A = \left[\begin{array}{c|c|c|c} D & 0 & 0 & 0 \\ I & I & -MI & 0 \\ 0 & 0 & I & I \end{array}\right].$$

Let $x = (y_{11}, y_{12}, \ldots, y_{mn}, a_{11}, \ldots, a_{mn}, z_{11}, \ldots, z_{mn}, b_{11}, \ldots, b_{mn})$. Consider the following set of binomials in $k[x]$.

$$G = \left\{ \begin{array}{ll} \underline{y_{ip} a_{iq}} - y_{iq} a_{ip}, & i = 1, \ldots, n;\ 1 \leqslant p < q \leqslant m \\ \underline{b_{ij}} - a_{ij}^{M_i} z_{ij}, & i = 1, \ldots, n;\ j = 1, \ldots, m \end{array} \right\}.$$

By Observation 2, G is contained in J. We show below that G can be used as a starting basis of J by showing that G is in fact a Gröbner basis of J. Let y denote the set of variables $(y_{11},\dots,y_{mn})$ and similarly a, z and b denote the other sets of variables. Then $x=(y, a, z, b)$ and the underlying polynomial ring is $k[x]=k[y, a, z, b]$. Let $>$ be a term order on $k[x]$ such that $b>y>z>a$. Within a block, the variables are sorted lexicographically according to index.

Theorem 4. *The set G is a Gröbner basis for J with respect to the term order $>$.*

Proof. Notice first that the initial term of every binomial in G with respect to $>$ is the underlined term. It is enough to show that for every binomial $x^\alpha - x^\beta \in J$, with initial term x^α, there is some g in G whose initial term divides x^α. Observation 2 implies that $x^\alpha - x^\beta$ is in J if and only if $\alpha-\beta\in S=\{t\in\mathbb{Z}^n\colon At=0\}$. We denote an element t in S by $t=(t_y, t_a, t_z, t_b)$ to indicate the correspondence between components of t and the columns of A. We denote the components of t_y by $(Y_{11},\dots,Y_{mn})$ and similarly for the others. We classify the elements in S in the following manner.

(a) Let $K_1=\{t\in S\colon t_y=t_b=0\}$. Now $t\in K_1$ if and only if $(t_a, t_z)\in\mathbb{Z}^{2mn}$ belongs to the lattice $S'=\{w\in\mathbb{Z}^{2mn}\colon A'w=0\}$ where

$$A'=\left[\begin{array}{c|c} I & -MI \\ 0 & I\end{array}\right].$$

But $S'=\{0\}$ since A' is an invertible square matrix. Therefore $K_1=\{0\}$. This implies that there is no binomial of the form $x^\alpha - x^\beta$ in J involving only the variables a_{ij} and z_{ij}.

(b) Let $K_2=\{t\in S\colon t_b=0\}$. Again $t\in K_2$ if and only if $(t_y, t_a, t_z)\in\mathbb{Z}^{3mn}$ belongs to the lattice $S''=\{w\in\mathbb{Z}^{3mn}\colon A''w=0\}$ where

$$A''=\left[\begin{array}{c|c|c} D & 0 & 0 \\ I & I & -MI \\ 0 & 0 & I\end{array}\right].$$

By (a) above, we may assume that $t_y\neq 0$. From the last mn rows of A'' we have that $t_z=0$. Therefore, $t\in K_2$ if and only if $(t_y, t_a)\in S'''=\{w\in\mathbb{Z}^{2mn}\colon A'''w=0\}$ where

$$A'''=\left[\begin{array}{c|c} D & 0 \\ I & I\end{array}\right].$$

Let Y_{ip} be the left most nonzero component of t_y. We may assume that $Y_{ip}>0$ since S is the set of integer points in a vector space which implies that it contains the negative of every element in it. The ith row of D in A''' implies that there exists some $q>p$ such that $Y_{iq}<0$. Therefore,

$$y_{ip} \text{ divides } x^{t^+} \quad\text{and}\quad y_{iq} \text{ divides } x^{t^-}.$$

Consider now the last mn rows of A'''. These rows imply that $A_{ip}=-Y_{ip}<0$ and $A_{iq}=-Y_{iq}>0$. Therefore,

$$y_{ip}a_{iq} \text{ divides } x^{t^+} \quad\text{and}\quad y_{iq}a_{ip} \text{ divides } x^{t^-}.$$

The initial term of $x^{t^+}-x^{t^-}$ with respect to $>$ is x^{t^+} since y_{ip} divides x^{t^+} and y_{ip} is the most expensive variable among those present in this binomial. But this implies that the initial term of $y_{ip}a_{iq} - y_{iq}a_{ip} \in G$ divides the initial term of $x^{t+} - x^{t-}$. Therefore, the initial term of all binomials associated with K_2 is divisible by the initial term of an element in G.

(c) Consider now a general element in $S = \{t \in \mathbb{Z}^n\colon At = 0\}$ with no variables restricted to be zero. By (a) and (b) above we may assume that $t_b \neq 0$. Let B_{rs} be the first nonzero component of t_b. As before, we may assume that $B_{rs} > 0$. Therefore,

$$b_{rs} \text{ divides } x^{t+}.$$

But x^{t+} is the initial term of $x^{t+} - x^{t-}$ under the term order being used, and this initial term is divisible by b_{rs}, which is the initial term of $b_{rs} - a_{rs}^{M_r} z_{rs} \in G$. Therefore $\mathit{init}_>(J) = \langle \mathit{in}_>(g),\ g \in G \rangle$ which proves that G is a Gröbner basis of J with respect to the term order $>$. □

The Buchberger algorithm is an iterative method that builds the Gröbner basis in steps from a starting basis of the ideal and a term order $>$. At every step it tests if two members of the current partial Gröbner basis can interact to give a new member whose initial term cannot be divided by the initial terms of all members currently present. If such a new member is generated, it is added to the current set of elements and the test continued until no new elements arise. The resulting set is a Gröbner basis with respect to $>$. We now restate a criterion from [1] which recognizes when two elements from the partial Gröbner basis cannot interact to give a new element during the algorithm.

Lemma 3 (Buchberger [1]). *If the initial terms of two polynomials f, g in the current partial Gröbner basis are relatively prime (i.e., have no common terms), then f and g cannot interact to generate a new element of the Gröbner basis.*

We now go back to the starting basis G we proposed and show that we can speed up the computation considerably by a decomposition idea along with an application of Lemma 3. Since G is a Gröbner basis of J, it can be used as a starting basis for the Buchberger algorithm. We now draw attention to a special feature of this starting basis. We can group the elements of G according to job types in the following way. With job i we associate the set G_i consisting of all elements of G whose variables have i as its first index.

$$G_i = \left\{ \begin{matrix} y_{ip}a_{iq} - y_{iq}a_{ip}, & 1 \leqslant p < q \leqslant m \\ b_{ij} - a_{ij}^{M_i} z_{ij}, & j = 1, \ldots, m \end{matrix} \right\} \subset G.$$

Therefore, $G = \bigcup_{i=1}^{n} G_i$. Consider the ideal J_i generated by the elements of G_i. Clearly $J_i \subseteq J$. We compute the reduced Gröbner basis of J_i with respect to the same cost on the variables as before, by using G_i as the starting basis. Let GB_i be the corresponding reduced Gröbner basis. Notice that the elements of each G_i involve only a subset of the complete set of variables. Also two distinct sets G_i and G_j have no variables in common. We now compute the reduced Gröbner basis for each such ideal J_i as i ranges over all the jobs. Let $GB = \bigcup_{i=1}^{m} GB_i$.

Corollary 3. *The set $GB = \bigcup_{i=1}^{m} GB_i$ is the reduced Gröbner basis $\mathscr{G}_{>_c}$ of the ideal J.*

Proof. Surely $GB = \bigcup_{i=1}^{m} GB_i$ is a basis of J since for each i, GB_i and G_i generate the same ideal J_i and $G = \bigcup_{i=1}^{m} G_i$ was a basis of J. Suppose GB was the starting basis in the Buchberger algorithm used to compute the reduced Gröbner basis of J with respect to the order $>_c$. For each job type i, the reduced Gröbner basis GB_i already contains all new elements that came out of the interaction between two elements in G_i. We now prove that a binomial from the set GB_i cannot interact with a binomial from a different set GB_j, to give a new element during the algorithm.

Consider the ideal J_i associated with job i, its starting basis G_i and reduced Gröbner basis GB_i. Since GB_i was obtained from G_i, its elements involve the same variables as those occurring in G_i. We now consider a different job j and the associated reduced Gröbner basis GB_j. If $f \in GB_i$ and $g \in GB_j$ then they have no common variables since they came from G_i and G_j which had no common variables. In particular, the initial terms of f and g with respect to any term order are relatively prime. By Lemma 3, f and g cannot interact in the Buchberger process to give a new member of the final Gröbner basis. Therefore $GB = \mathscr{G}_{>_c}$. □

The corollary above shows that we can decompose the starting basis according to jobs and compute the reduced Gröbner basis of each piece separately and then concatenate these reduced Gröbner bases to get the overall Gröbner basis (test set) we are looking for. This decomposition results in tremendous increase in the speed of the computations involved. We believe that identification of problem specific starting bases and its decomposition whenever possible should be a general strategy in solving large problems. In the next section we present some of our computational results as well as comparisons of this method with the general Conti–Traverso algorithm for the specific class of matrices associated with the reduced IP.

3.4. An example

In this section we illustrate the procedures described earlier using an example of scheduling a job on 2 machines. Thus $m = 2$ and $n = 1$. Let the production costs per unit be $L'_{11} = 3$ and $L'_{12} = 4$ and the setup cost per batch be $K_{11} = 5$ and $K_{12} = 3$. The machines are identical with a capacity of 10, the processing times and setup times are $p_{11} = p_{12} = 1$ and $S_{11} = S_{12} = 1$ respectively and $M_1 = 3$. We are interested in finding the optimal schedule for $\gamma = 0.6$ using 100 samples of demand drawn from a normal distribution with a mean of 12 and a variance of 6.

This instance corresponds to the following:

minimize $5z_{11} + 3z_{12} + 12y_{11} + 16y_{12}$

such that:

$y_{11} + y_{12} = 3$

$y_{11} - 3z_{11} \leqslant 0$

$$y_{12} - 3z_{12} \leqslant 0$$

$$4y_{12} + z_{11} \leqslant 10$$

$$4y_{12} + z_{12} \leqslant 10$$

$$\text{Prob}\{D_1 \tfrac{1}{3} y_{11} + z_{11} \leqslant 10,\ D_1 \tfrac{1}{3} y_{12} + z_{12} \leqslant 10\} \geqslant 0.6$$

$$z_{ij} \in \{0, 1\}, \quad y_{ij} \in \{0, 1, 2, 3\}.$$

3.4.1. Gröbner directions

In an 8-dimensional space $(y_{11}, y_{12}, a_{11}, a_{12}, z_{11}, z_{12}, b_{11}, b_{12})$, the three elements of the starting basis for this problem as described in Section 3.3 are:

$$y_{11}a_{12} - y_{12}a_{11},$$

$$a_{11}^3 z_{11} - b_{11},$$

$$a_{12}^3 z_{12} - b_{12}.$$

The six elements in the Gröbner basis (test directions) obtained using *Macaulay* are:
(1) $y_{12}a_{11} - y_{11}a_{12}$,
(2) $y_{12}^2 b_{11} - z_{11} y_{11}^2 a_{11} a_{12}^2$,
(3) $z_{12} a_{12}^3 - b_{12}$,
(4) $z_{12} y_{12} a_{12}^2 b_{11} - z_{11} y_{11} a_{11}^2 b_{12}$,
(5) $z_{11} a_{11}^3 - b_{11}$, and
(6) $z_{11} y_{11} a_{11}^2 a_{12} - y_{12} b_{11}$.
Note that $y_{12}a_{11} - y_{11}a_{12}$ is interpreted as the direction from (1, 0, 0, 1, 0, 0, 0, 0) to (0, 1, 1, 0, 0, 0, 0, 0).

3.4.2. Walk back procedure: An illustration

The optimal solution to the reduced IP (Section 2.3) is $y_{11}^{\mathrm{r}} = 3$ and $z_{11}^{\mathrm{r}} = 1$ (other y_{ij}, z_{ij} are zero) with an objective function value of 41. This solution is not feasible for (P1) because it violates constraint (3).

Fig. 2 illustrates the walk back procedure described in Section 3.1. Recall that constraints (3) and (4) are checked using a membership oracle; the details are described in Section 4.1. It should be noted that whenever we branch along a particular direction we also evaluate the nonnegativity of the variables in the new solution. If the solution violates nonnegativity constraints we prune the branch. (Thus we prune the branch along direction 1 from the reduced IP solution.) Otherwise, the membership oracle is queried for feasibility with respect to constraints (3) and (4). If the oracle returns a negative answer, we continue with the branching (as illustrated by branching along direction 3 from the reduced IP solution). If we obtain a feasible solution to (P1) the branch is pruned. In this example, from the reduced IP solution, we first branch along direction 3 and then along direction 1. Since all the paths in Fig. 2 have been pruned, the solution obtained is optimal. The oracle is set up in such a way so as to provide the highest γ for

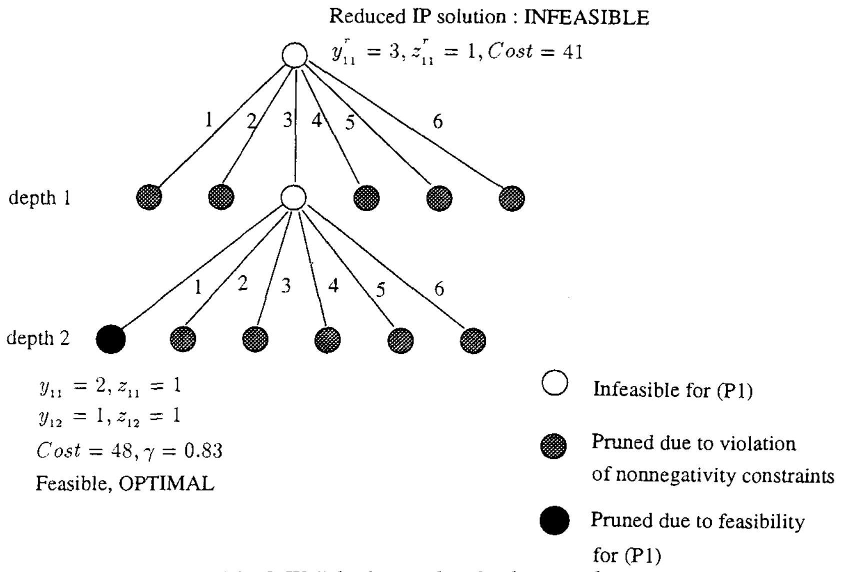
Reduced IP solution : INFEASIBLE
$y^r_{11} = 3, z^r_{11} = 1, Cost = 41$
1
2
3
4
5
6
depth 1
1
2
3
4
5
6
depth 2
$y_{11} = 2, z_{11} = 1$
$y_{12} = 1, z_{12} = 1$
$Cost = 48, \gamma = 0.83$
Feasible, OPTIMAL
Infeasible for (P1)
Pruned due to violation of nonnegativity constraints
Pruned due to feasibility for (P1)

Fig. 2. Walk back procedure for the example.

which this solution remains optimal; in this case it is 0.83. For higher values of γ we continue branching beyond depth 2.

4. Computational experience

4.1. The membership oracle

The membership oracle is constructed after the optimal solution to the reduced IP is found. Let $\alpha^r = (y^r_{ij}, z^r_{ij})$ be the optimal solution to the reduced IP (from Section 2.3). Whenever the empirical demands are available, we use them directly; otherwise, samples from the demand distribution are generated. The number of data points or samples, S, required depends on errors ϵ_1, ϵ_2 specified; see the next subsection for the exact relationship. Let $(d^k_1, \ldots, d^k_n)$, $k = 1, \ldots, S$, be the samples. The membership oracle is a table with m columns (one for each machine) and $S+1$ rows (R^r_{pq}, $p = 1, \ldots, S+1$, $q = 1, \ldots, m$). Each element in the first S rows represents the *slack* in machine j for the sample corresponding to that row for the solution (y^r_{ij}, z^r_{ij}). Thus, $R^r_{pq} = C_q - \sum^n_{i=1}((y^r_{iq}/M_i)d^p_i + z^r_{iq}S_{iq})$, for $p = 1, \ldots, S$, $q = 1, \ldots, m$. The $(S+1)$st row stores the slack in the complicated constraint of the IP, $C_q - \sum^n_{i=1}((y^r_{iq}/M_i)\hat{D}_i + z^r_{iq}S_{iq})$. Some of the elements in the table can be negative, implying that more than the available capacity has been used up for production and setups. Note that α^r is feasible for problem (P1) if $(S+1)$st row elements of R^r are all nonnegative, and there are at least γS other rows of R^r that have all nonnegative elements.

If α^r is not feasible for (P1), we walk onto other points using a *walk back* procedure. The walk back procedure is a variable depth procedure, starting from

$(y^{\mathrm{r}}_{ij}, z^{\mathrm{r}}_{ij})$. The implementation requires: (1) a recursive call to a subroutine that moves one more step into the polyhedron in a test direction and (2) a three-dimensional array (T) that stores the change in each of the elements of R^{r} due to each test direction. Let a test direction be $g_{\mathrm{t}} = (y^{\mathrm{t}}_{ij}, z^{\mathrm{t}}_{ij})$. Then T_{pqt} equals $\Sigma^{n}_{i=1}((y^{\mathrm{t}}_{iq}/M_i)d^{p}_{i} + z^{\mathrm{t}}_{iq}S_{iq})$ for $p = 1,\dots,S$; $q = 1,\dots,m$ and $T_{S+1qt} = \Sigma^{n}_{i=1}((y^{\mathrm{r}}_{iq}/M_i)\hat{D}_i + z^{\mathrm{t}}_{iq}S_{iq})$ for $q = 1,\dots,m$.

At each call, the feasibility of any point (α) is determined as follows. Let R^{α} be a table for α analogous to R^{r}. We do not store this table but rather compute the table dynamically using R^{r} and T. A point α is feasible for problem (P1) if $(S+1)$st row elements of R^{α} are all nonnegative, and there are at least γS other rows of R^{α} that have all elements nonnegative. To construct R^{α}, let the path to α from α^{r} be obtained by test directions (not necessarily distinct) $g_{\mathrm{t}_1},\dots,g_{\mathrm{t}_f}$. Then, $R^{\alpha}_{pq} = R^{\mathrm{r}}_{pq} + \Sigma^{f}_{i=1}T_{pqi}$ for $p = 1,\dots,S+1$, $q = 1,\dots,m$.

4.2. Number of samples

When the empirical data is not available, then we need to generate samples from the demand distribution, $F(x_1,\dots,x_n) = \text{Probability}\{D_1 \leqslant x_1,\dots,D_n \leqslant x_n\}$. We use the following result to determine the number of samples required.

Theorem 5 (Hoeffding [5]). *Let $S_k = X_1 + \cdots + X_k$, where the X_i are independent random variables with mean m and with $0 \leqslant X_i \leqslant P$ for some P. Then*

$$Pr\left[\left|\frac{S_k}{k} - m\right| \geqslant \epsilon_1\right] \leqslant 2\mathrm{e}^{-k(\epsilon_1^2/P^2)}.$$

In our setting, $X_i = \mathscr{I}(\{\Sigma^{n}_{i=1} y_{ij}p_{ij}(D_i/M_i) \leqslant x + C_j - \Sigma^{n}_{i=1} z_{ij}S_{ij}, \forall j\})$ where $\mathscr{I}$ is an indicator function. Clearly, $m = \gamma$, and $P = 1$. Given ϵ_1 and ϵ_2, we need $S = -(1/2\epsilon_1^2)\log\frac{1}{2}\epsilon_2$ samples.

4.3. Standard starting bases

As mentioned in Section 3, we were motivated to look for an improved starting basis because the following available (standard) starting bases were not computationally attractive.

Conti and Traverso describe a procedure to compute the reduced Gröbner basis, $>_c$ of a general integer program:

$$\begin{array}{lll}(\text{IP}) & \min & cx \\ & \text{s.t.} & Ax = b \\ & & x \in \mathbb{N}^n.\end{array}$$

The method first introduces the extra variables $y_1,\dots,y_m$ (one variable per row of A) and a variable t. Let $I = \langle y^{a_i^+} - y^{a_i^-}x_i, (ty_1 \cdots y_m) - 1\rangle$. The reduced Gröbner basis of I is calculated with respect to an order $>$ such that (1) $y, t > x$ and (2) $x^{\alpha} > x^{\beta}$ if and only if $x^{\alpha} >_c x^{\beta}$. The theory of Gröbner bases implies that the elements of the resulting reduced Gröbner basis containing only the x-variables constitute $G_{>_c}$. This starting basis failed to produce a Gröbner basis even for a two-machine–three-job problem in a *SPARC Station* 2 with 14 Megs of RAM.

A more sophisticated method again due to Conti and Traverso is as follows. Let $B = \{b_1, \ldots, b_p\} \subset \mathbb{Z}^n$ be a $\mathbb{Z}$-basis [8] of the lattice $L = \{x \in \mathbb{Z}^n\colon\ Ax = 0\}$. Let $I = \langle x^{b_i^+} - x^{b_i^-}, (tx_1 \cdots x_n) - 1 \rangle$. Again we compute the reduced Gröbner basis of I with respect to the order $>$ such that (1) $t > x$ and (2) $x^\alpha > x^\beta$ if and only if $x^\alpha >_c x^\beta$. As before, those elements of the resulting Gröbner basis containing only the x-variables constitute $G_{>_c}$. Note that t is still required, and its presence not only increases the running time for a single job case but also precludes the decomposition by jobs in a multiple job setting. This starting basis was computationally feasible only for problems with less than four machines; however, even with one, two or three machines the starting basis of Section 3.3 was clearly superior.

4.4. Hot start

We notice that the reduced IP decomposes by jobs and if M_i are all equal, the n subproblems are similar except for the costs. Thus, as an alternative to starting all the subproblems with generators of Section 3.3, we can start the first subproblem with starting generators of Section 3.3 and subproblems $i = 2, \ldots, n$ with the Gröbner basis of subproblem $i - 1$. However, this method did not lead to any improvement in running times.

4.5. Numerical results

Computational results for three problems are provided below. For simplicity, we assume that all M_i are the same. The first problem has three machines and four jobs. Here we vary M, capacities and covariance structure to demonstrate the logic of our algorithm as well as provide some basic insights into solutions of these type of problems. The second problem has three machines and six jobs, and we vary the demand variance and covariance, the costs, and M. The third problem based on a real word situation, from a laminate manufacturer, has four machines and seven products. (An ABC analysis of demand at this plant showed that these seven products accounted for over 85% of total demand.) All numbers in the third problem are rescaled for proprietary reasons while maintaining their relative values.

All experiments were conducted on a *SPARC Station* 2. As stated earlier, the walk back procedure is a variable depth procedure. The membership oracle is designed such that only those γ values that cause a change in optimal solution are used in solving problems. For example, while computing the optimal solution at $\gamma = 0$, we will know until what level of γ this solution will remain optimal. In Tables 1–7, γ represents the highest value at which the solution presented is optimal. If optimality could not be

	job 1	job 2	job 3	job 4	job 5	job 6	job 7
machine 1	3, 5	4, 3	3, 3	2, 1	3, 1	3, 3	4, 5
machine 2	4, 1	4, 1	2, 4	2, 5	4, 3	3, 1	2, 2
machine 3	3, 4	3, 2	2, 5	2, 4	4, 4	2, 2	4, 4
machine 4	3, 5	4, 4	2, 4	3, 4	3, 1	4, 5	2, 2

Fig. 3. Setup and production costs for Example 4: K_{ij}, L'_{ij}.

shown within a depth of eight or two hundred minutes of CPU time, we report the best feasible solution found.

Example 2. Here $n = 4$ and $m = 3$. Tables 1–2 shows how the the optimal solutions change with γ for a simple example. In experiments 1–2, the capacities are (25, 25, 25) and the demands are independent. In experiment 3–4, the capacities are changed to (28, 20, 28). In experiments 5–8, the capacities are (20, 20, 35); in experiment 8 a positive covariance is introduced between products one and four, and between products two and three. 250 samples are used in the membership oracle.

Example 3. Tables 3–6 shows changes in optimal solutions as we vary the capacities, M and the covariance structure when $n = 6$ and $m = 3$. 250 samples are used in the membership oracle.

Example 4. Table 7 contains the solution for a problem from the real world that has four machines and seven jobs. The data is as follows. The capacities on the machines are (20, 20, 20, 40) respectively. All setup times and production times are 1. The setup and production costs are provided in Fig. 3: (K_{ij}, L'_{ij}). The mean demands are (20, 16, 12, 8, 8, 4, 4); the variance–covariance matrix is provided in Fig. 4.

Based on the computational experiments, we can summarize the following.

(1) The starting generators of Section 3.3, which are input to *Macaulay*, provide the directions quickly, typically in less than *one second*.

(2) The time to answer a membership query is also very small. The entire walk back

$$\begin{pmatrix} 36 & 0 & -10.8 & 0 & 0 & 0 & 7.34 \\ 0 & 23 & 0 & 5.9 & 11.5 & 0 & 0 \\ -10.8 & 0 & 13 & 0 & 0 & -4.4 & -4.4 \\ 0 & 5.9 & 0 & 6 & 0 & 0 & 0 \\ 0 & 11.5 & 0 & 0 & 23 & 0 & 0 \\ 0 & 0 & -4.4 & 0 & 0 & 6 & 0 \\ 7.34 & 0 & -4.4 & 0 & 0 & 0 & 6 \end{pmatrix}$$

Fig. 4. Variance–covariance matrix for Example 4.

Table 1
Results for Example 2: Changing M and capacities

Experiment 1	$M=1$	$\gamma=0.512$	$\gamma=0.752$	$\gamma=0.996$	
	Product 1	$y_{12}=1$	$y_{11}=1$	$y_{11}=1$	
	Product 2	$y_{23}=1$	$y_{23}=1$	$y_{23}=1$	
	Product 3	$y_{33}=1$	$y_{33}=1$	$y_{32}=1$	
	Product 4	$y_{42}=1$	$y_{42}=1$	$y_{42}=1$	
	Time (secs)	0.5	0.5	0.6	
Experiment 2	$M=2$	$\gamma=0.512$	$\gamma=0.752$	$\gamma=0.996$	
	Product 1	$y_{12}=2$	$y_{11}=2$	$y_{11}=2$	
	Product 2	$y_{23}=2$	$y_{23}=2$	$y_{23}=2$	
	Product 3	$y_{33}=2$	$y_{33}=2$	$y_{32}=2$	
	Product 4	$y_{42}=2$	$y_{42}=2$	$y_{42}=2$	
	Time (secs)	0.7	0.6	0.7	
Experiment 3	$M=2$	$\gamma=0.896$	$\gamma=0.972$	$\gamma=0.992$	
	Product 1	$y_{11}=2$	$y_{11}=2$	$y_{11}=2$	
	Product 2	$y_{23}=2$	$y_{21}=1$; $y_{23}=1$	$y_{22}=2$	
	Product 3	$y_{33}=2$	$y_{33}=2$	$y_{32}=1$; $y_{33}=1$	
	Product 4	$y_{42}=2$	$y_{42}=2$	$y_{42}=2$	
	Time (secs)	0.6	2.0	2.1	
Experiment 4	$M=4$	$\gamma=0.896$	$\gamma=0.912$	$\gamma=0.988$	$\gamma=0.996$
	Product 1	$y_{11}=4$	$y_{11}=3$; $y_{12}=1$	$y_{11}=4$	$y_{11}=3$; $y_{12}=1$
	Product 2	$y_{23}=4$	$y_{21}=4$	$y_{21}=1$; $y_{23}=3$	$y_{21}=1$; $y_{23}=3$
	Product 3	$y_{33}=4$	$y_{33}=4$	$y_{33}=4$	$y_{32}=4$
	Product 4	$y_{42}=4$	$y_{42}=4$	$y_{42}=4$	$y_{42}=4$
	Time (secs)	0.9	2.3	2.3	18.1

procedure may take a long time because many points need to be evaluated, and not because the evaluation at any single point is time consuming (also see item (5)).

(3) At lower values of γ, the optimal solution can be found relatively quickly.

(4) As M_i is doubled, a finer partition into level sets is possible, as well as the highest γ achievable increases (see Table 1, for example). However, this may not always happen. The level sets for $M=4$ may sometimes equal those in $M=2$ (see Table 2, for example), and so not taking advantage of extra flexibility provided for scheduling. Similarly, a comparison between $M=1$ and $M=2$ shows no pattern that can be expected a priori (for example, see Tables 1–2).

(5) $M=3$ can achieve higher γ than $M=4$. This is because a $\frac{1}{2}$, $\frac{2}{3}$ split of a job may result in a lower shortfall probability than a $\frac{1}{4}$, $\frac{3}{4}$ split or a $\frac{2}{4}$, $\frac{2}{4}$ split.

(6) Typically a good feasible solution at any γ can be found quickly. The time taken to find just a feasible solution is even shorter. In Example 4, for instance, a solution of cost 220 that has a γ of 0.94 was found in 721.9 seconds (not shown in Table 7): $y_{12}=y_{13}=1$; $y_{23}=y_{34}=2$; $y_{41}=y_{54}=2$; $y_{61}=y_{74}=2$. Showing that it is optimal or finding a better solution takes longer as more paths need to be pruned. A good lower bounding technique should help here and allow the solution of larger problems.

Table 2
Results for Example 2 (continued): changing M, capacities and covariance

Experiment 5	$M=2$	$\gamma=0.964$	$\gamma=0.996$	
	Product 1	$y_{11}=2$	$y_{11}=1=y_{12}$	
	Product 2	$y_{23}=2$	$y_{23}=2$	
	Product 3	$y_{33}=2$	$y_{33}=2$	
	Product 4	$y_{42}=2$	$y_{41}=1=y_{42}$	
	Time (secs)	0.7	12.5	
Experiment 6	$M=3$	$\gamma=0.964$	$\gamma=0.984$	
	Product 1	$y_{11}=3$	$y_{11}=2; y_{12}=1$	
	Product 2	$y_{23}=3$	$y_{23}=3$	
	Product 3	$y_{33}=3$	$y_{33}=3$	
	Product 4	$y_{42}=3$	$y_{42}=3$	
	Time (secs)	0.8	0.8	
Experiment 7	$M=4$	$\gamma=0.964$	$\gamma=0.996$	
	Product 1	$y_{11}=4$	$y_{11}=3; y_{12}=1$	
	Product 2	$y_{23}=4$	$y_{23}=4$	
	Product 3	$y_{33}=4$	$y_{33}=4$	
	Product 4	$y_{42}=4$	$y_{42}=4$	
	Time (secs)	0.8	0.9	
Experiment 8	$M=2$	$\gamma=0.96$	$\gamma=0.988$	$\gamma=1$
	Product 1	$y_{11}=2$	$y_{12}=2$	$y_{12}=2$
	Product 2	$y_{23}=2$	$y_{23}=2$	$y_{21}=1=y_{23}$
	Product 3	$y_{33}=2$	$y_{33}=2$	$y_{33}=2$
	Product 4	$y_{42}=2$	$y_{41}=2$	$y_{41}=2$
	Time (secs)	0.6	0.7	3.2

Table 3
Results for Example 3: 3 machines, 6 jobs, $M=2$, randomly generated costs, 250 samples

γ	Objective value	Solution	Time (secs)	Depth found	Optimal
0.728	185	$y_{13}=y_{22}=2$ $y_{32}=y_{41}=2$ $y_{51}=y_{63}=2$	0.2	0	Yes
0.92	186	$y_{13}=y_{22}=2$ $y_{32}=y_{43}=2$ $y_{51}=y_{63}=2$	0.4	1	Yes
0.956	194	$y_{13}=y_{22}=2$ $y_{33}=y_{32}=1$ $y_{41}=y_{43}=1$ $y_{51}=y_{63}=2$	3	3	No
0.968	195	$y_{22}=y_{13}=2$ $y_{31}=y_{32}=1$ $y_{43}=2$ $y_{51}=y_{63}=2$	4.8	3	No
0.992	202	$y_{13}=y_{22}=2$ $y_{32}=y_{33}=1$ $y_{43}=y_{51}=2$ $y_{61}=y_{63}=1$	4.8	3	No

Table 4
Results for Example 3 (continued): 3 machines, 6 jobs, $M = 3$, randomly generated costs, 250 samples

γ	Objective value	Solution	Time (secs)	Depth found	Optimal
0.728	185	$y_{13} = y_{22} = 3$ $y_{32} = y_{41} = 3$ $y_{51} = y_{63} = 3$	0.3	0	Yes
0.92	186	$y_{13} = y_{22} = 3$ $y_{32} = y_{43} = 3$ $y_{51} = y_{63} = 3$	0.4	1	Yes
0.936	187	$y_{13} = y_{22} = 3$ $y_{32} = 3$ $y_{41} = 1,\ y_{43} = 2$ $y_{51} = y_{63} = 3$	0.4	1	Yes
0.972	192	$y_{13} = y_{22} = 3$ $y_{32} = 2,\ y_{33} = 1$ $y_{41} = 1,\ y_{43} = 2$ $y_{51} = y_{63} = 3$	4.1	3	No
0.976	193	$y_{13} = 3$ $y_{21} = 1,\ y_{22} = 2$ $y_{32} = y_{43} = 3$ $y_{51} = y_{63} = 3$	3.5	3	No
0.992	198	$y_{13} = y_{22} = 3$ $y_{32} = 2,\ y_{33} = 1$ $y_{43} = y_{51} = 3$ $y_{61} = 1,\ y_{63} = 2$	731	5	No
1.0	200	$y_{13} = 3$ $y_{22} = 2,\ y_{23} = 1$ $y_{32} = 3$ $y_{43} = y_{51} = 3$ $y_{61} = 1,\ y_{63} = 2$	731	5	No

(7) Although optimal solutions were not found at higher values of γ for Examples 3–4, our results show that we can find good solution in reasonable time. To see this, in Example 3, all solutions were within 10% of optimality; in Example 4, all solutions were within 5.5%. Note that for Example 2, all solutions shown are optimal.

(8) If a feasible solution is not found within depth five, it is a strong signal that the problem is infeasible. A shortcoming of this approach, a problem shared by other available methods for integer and chance-constrained programming, is that infeasibility cannot be detected easily.

(9) The lot splitting on some of the optimal solutions can be explained intuitively. However, in most cases the optimal (or good) solutions cannot be predicted a priori.

Table 5
Results for Example 3 (continued): 3 machines, 6 jobs, $M = 4$, randomly generated costs, 250 samples

γ	Objective value	Solution	Time (secs)	Depth found	Optimal
0.728	185	$y_{13} = y_{22} = 4$ $y_{32} = y_{41} = 4$ $y_{51} = y_{63} = 4$	0.4	0	Yes
0.92	186	$y_{13} = y_{22} = 4$ $y_{32} = y_{43} = 4$ $y_{51} = y_{63} = 4$	0.5	1	Yes
0.94	187	$y_{13} = y_{22} = 4$ $y_{32} = 4$ $y_{41} = 1$, $y_{43} = 3$ $y_{51} = y_{63} = 3$	1.8	1	Yes
0.98	191	$y_{13} = y_{22} = 4$ $y_{32} = 3$, $y_{33} = 1$ $y_{41} = 1$, $y_{43} = 3$ $y_{51} = y_{63} = 3$	4.4	3	No
0.988	198	$y_{13} = 4$ $y_{22} = 3$, $y_{23} = 1$ $y_{43} = y_{51} = 4$ $y_{61} = 1$, $y_{63} = 3$	984	5	No
0.996	199	$y_{13} = y_{22} = 4$ $y_{32} = 3$, $y_{33} = 1$ $y_{43} = 4$ $y_{51} = 2$, $y_{53} = 2$ $y_{63} = 4$	1603	5	No

Note that (P1) with $\gamma = 0$ is an integer programming problem. Thus, our approach can be considered as a decomposition method to solve integer programs.

5. Conclusions

Another application of our framework arises in the design of wafers in semi-conductor fabrication.

5.1. An application in semi-conductor wafer design

Consider the following problem of chip allocation to wafers:

$$\text{minimize} \quad \sum_i \sum_j K_{ij} Z_{ij}$$

such that:

$$Y_{ij} = CZ_{ij} \quad \forall i, j$$

Table 6
Results for Example 3 (continued): 3 machines, 6 jobs, $M = 4$, randomly generated costs, jobs 1, 2, 3 are positively correlated, jobs 4, 5, 6 are negatively correlated with correlation coefficient = 0.5; 250 samples

γ	Objective value	Solution	time (secs)	Depth found	Optimal
0.732	185	$y_{13} = y_{22} = 4$ $y_{32} = y_{41} = 4$ $y_{51} = y_{63} = 4$	0.4	0	Yes
0.904	186	$y_{13} = y_{22} = 4$ $y_{32} = y_{43} = 4$ $y_{51} = y_{63} = 4$	0.5	1	Yes
0.94	190	$y_{13} = y_{22} = 4$ $y_{32} = 3,\ y_{33} = 1$ $y_{43} = 4$ $y_{51} = y_{63} = 3$	4.2	3	No
0.996	191	$y_{13} = y_{22} = 4$ $y_{32} = 3,\ y_{33} = 1$ $y_{41} = 2,\ y_{43} = 2$ $y_{51} = y_{63} = 4$	4.3	3	No
1.0	198	$y_{13} = y_{22} = 4$ $y_{32} = 3,\ y_{33} = 1$ $y_{41} = 3,\ y_{43} = 1$ $y_{51} = 3,\ y_{53} = 1$ $y_{63} = 4$	1710	5	No

$$\text{Probability}\left\{\min_i \sum_j \alpha_{ij} Y_{ij} \geqslant B\right\} \geqslant \gamma$$

$$Y_{ij} \in \mathbb{N}^m, \quad Z_{ij} \in \{0, 1\}.$$

Here j is an index for wafers, i an index for chip type, and B the number of *sets* of the chips desired. A setup cost is incurred due to increased number of chip types on a wafer, and C is the capacity of a wafer. The randomness is due to the fact that the whole wafer may be bad, or only certain fraction of chips on a wafer may be bad. This is captured by α_{ij}; note again that the feasible set may be nonconvex. The membership oracle for this problem can be developed easily following the structure described in Section 4.1. The results of this paper can then be applied to this problem with minor modifications.

5.2. Summary

We have modeled a manufacturing problem typical of industrial suppliers as a chance constrained integer program. We have provided a novel algorithm based an a very recent geometric interpretation of an algebraic approach originally developed to solve general integer programming problems. Our approach does not rely on linear relaxations and does not even require that the linear relaxation of the feasible set be convex. Our method

Table 7
Results for Example 4: 4 machines, 7 jobs, $M = 2$; $*$ means solution not proved optimal within time or depth constraint

γ	Objective Value	Solution	Time to find solution (secs)	Time to prove optimality (secs)
0.696	205	$y_{14} = y_{23} = 2$ $y_{32} = y_{41} = 2$ $y_{54} = y_{63} = 2$ $y_{74} = 2$	2.0	89.1
0.824	206	$y_{14} = y_{23} = 2$ $y_{32} = y_{41} = 2$ $y_{51} = y_{54} = 1$ $y_{63} = y_{74} = 2$	87.5	87.5
0.876	208	$y_{14} = y_{23} = 2$ $y_{32} = y_{43} = 2$ $y_{51} = y_{63} = 2$ $y_{74} = 2$	34.4	$*$
0.904	210	$y_{14} = y_{23} = 2$ $y_{32} = y_{33} = 1$ $y_{41} = y_{54} = 2$ $y_{63} = y_{72} = 2$	154.5	$*$
0.908	213	$y_{11} = y_{14} = 1$ $y_{23} = y_{32} = 2$ $y_{43} = y_{54} = 2$ $y_{63} = y_{74} = 2$	298.4	$*$
0.924	214	$y_{13} = y_{14} = 1$ $y_{23} = y_{32} = 2$ $y_{41} = y_{54} = 2$ $y_{61} = y_{74} = 2$	344.7	$*$
0.956	216	$y_{13} = y_{14} = 1$ $y_{23} = y_{34} = 2$ $y_{42} = y_{51} = 2$ $y_{62} = y_{74} = 2$	13321.8	$*$

requires only a membership oracle to store complicating and nonlinear constraints. Thus, this solution methodology is not limited to chance constrained programs and can be used to solve a very wide class of problems. We found a decomposition of the problem and a special starting basis that make this approach computationally feasible. To our knowledge, this is the first paper of this type; we believe that future research concentrating on similar decompositions and developing attractive starting bases will yield efficient algorithms for other problems modeled in this way.

Acknowledgements

The authors would like to thank Professor Bernd Sturmfels for several important inputs, both technical and editorial, to this paper.

References

[1] B. Buchberger, "Gröbner basis — an algorithmic method in polynomial ideal theory," in: N.K. Bose, ed., *Multidimensional Systems Theory* (Reidel, Dordrecht, 1985) Chapter 6.

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

[3] D. Cox, J. Little and D. O'Shea, *Ideals, Varieties and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra* (Springer, Berlin, 1992).

[4] M.A. Duran and I.F. Grossmann, "An outer approximation algorithm for a class of mixed integer nonlinear programs," *Mathematical Programming* 36 (3) (1986) 307–339.

[5] W. Hoeffding, "Probability inequalities for sums of bounded random variables, "*American Statistical Association Journal* (1963) 13–30.

[6] R. Kannan, J. Mount and S. Tayur, "A randomized algorithm to optimize over certain convex sets," *Mathematics of Operations Research*, to appear.

[7] H.E. Scarf, "Neighborhood systems for production sets with indivisibilities," *Econometrica* 54 (1986) 507–522.

[8] A. Schrijver, *Theory of Linear and Integer Programming*, Interscience Series in Discrete Mathematics and Optimization (Wiley, New York, 1986).

[9] M. Stillman, M. Stillman and D. Bayer, "Macaulay User Manual", Cornell University, Ithaca, NY, 1989.

[10] B. Sturmfels, "Gröbner bases of toric varieties," *Tohuku Mathematical Journal* 43 (1991) 249–261.

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

[12] R. Wets, "Stochastic programming", in: G.L. Nemhauser, A.H.G. Rinnooy Kan and M.J. Todd, eds., *Handbooks in Operations Research and Management Science*, Vol. 1 (North-Holland, Amsterdam, 1989).