# Parametric Integer Programming

Paul Feautrier

Laboratoire MASI, Institut Blaise Pascal
Université de Versailles St-Quentin
45 Avenue des Etats-Unis
78035 VERSAILLES CEDEX FRANCE

e-mail : `feautrier@masi.ibp.fr`

September 1988

**Abstract**

When analysing computer programs (especially numerical programs in which arrays are used extensively), one is often confronted with integer programming problems. These problems have three peculiarities:

- feasible points are ranked according to lexicographic order rather than the usual linear economic function;
- the feasible set depends on integer parameters;
- one is interested only in exact solutions.

The difficulty is somewhat alleviated by the fact that problems sizes are usually quite small. In this paper we show that:

- the classical simplex algorithm has no difficulty in handling lexicographic ordering;
- the algorithm may be executed in symbolic mode, thus giving the solution of continuous parametric problems;
- the method may be extended to problems in integers.

We prove that the resulting algorithm always terminate and give an estimate of its complexity.

**Résumé**

L'analyse sémantique des programmes (spécialement des programmes numériques utilisant des tableaux), conduit à la résolution de problèmes de programmation numérique en nombres entiers. Ces problèmes ont trois particularités:

- les points faisables ne sont pas classés suivant une fonction économique linéaire, mais suivant l'ordre lexicographique;
- le problème dépend de paramètres, eux aussi entiers;
- seules les solutions exactes sont intéressantes.

En compensation, la taille des problèmes à traiter est faible; il est envisageable de rechercher une solution complète. Dans ce papier, nous montrons:

- que l'algorithme classique du simplexe s'adapte sans difficulté au traitement de l'ordre lexicographique;
- qu'il est possible de l'exécuter symboliquement pour obtenir la solution de problèmes paramétriques continus;
- que cette technique s'étend à la résolution de problèmes en nombres entiers.

On prouve la convergence de l'algorithme ainsi obtenu et on donne une idée de sa complexité.

# 1 Introduction

When analyzing computer programs in which arrays are used, one often has to solve parametric integer problems. Consider for instance the following (somewhat contrived) piece of code:

```
for i := 0 to m do
   for j := 0 to n do
      a [2*i+j] := i+j;
```

After execution of these `for` loops, for which values of `k` is `a[k]` defined? If so, what is its value? To answer these questions, note first that `a[k]` is assigned a value for all pairs $(i, j)$ such that:

$$\begin{array}{rcl} 0 \leq & i & \leq m \\ 0 \leq & j & \leq n, \\ 2i + j & = & k. \end{array}$$

Furthermore, the definitive value of `a[k]` is given by the latest such access. Since the temporal sequence of accesses is given by the lexical ordering of $(i, j)$, this imply that:

$$\texttt{a [k]} = i_{max} + j_{max},$$

where $(i_{max}, j_{max})$ is the lexical maximum of the set:

$$\mathsf{F}(k, m, n) = \{(i, j) | 0 \leq i \leq m, 0 \leq j \leq n, 2i + j = k\}.$$

Among other things, `a[k]` is undefined when $\mathsf{F}(k, m, n)$ is empty. While one sees immediately that this occurs as soon as:

$$k > 2m + n,$$

finding the proper value of $i_{max}$ and $j_{max}$ is by no means easy.

Generalizing from the above exemple and similar riddles, we are lead to a study of the following problem:

- We are given a finite set of linear inequalities in a set of variables and parameters.
- Both variables and parameters are restricted to positive integer values.
- We are required to find the lexical minimum of the feasible set (the set of variable values which satisfies the given inequalities), as a function of the parameters.

Since there are various devices for replacing equalities by inequalities, we may suppose that the only constraints are inequalities of the $\geq 0$ type. The motivation of the change from lexical maximum to minimum lies in the fact that lexical ordering is well founded. Hence the lexical minimum of sets such as $\mathsf{F}$ above always exists, which is not true for the lexical maximum. In cases of program semantics interest, there is always an upper bound in evidence for each variable, and the change, if necessary, is easily done.

There are other differences with the problems one usually encounter in operation research. The problems are quite small. The unknown count is related to the maximum loop nesting, while the equation count is related to the array dimension. Both these quantities are small integers. In operation research, the elements of the feasible set are ranked according to some linear economic function. The lexicographic rule was introduced by Dantzig, Orden and Wolfe in 1954 as a mean to prevent cycling in the case of degeneracy: see [Dan63]. In the model we are interested in, lexicographic ordering replaces the classical linear economic function. It is a striking fact that the same algorithm (basically Dantzig's Simplex) gives the solution in both cases.

Another important difference is that we are not interested in approximate solutions. The information we gather will be used for restructuring programs, and such transformations must be based on exact data in order not to introduce errors.

The balance of the paper is dedicated to the construction and proof of a parametric integer programming algorithm. In paragraph 2, we will review the classical continuous non parametric simplex algorithm. Paragraph 3 will extend this algorithm to the parametric case. The resulting technique is equivalent to an algorithm of [Gal79], albeit much simpler to understand and to implement. In this paragraph we will introduce the concept of a problem tree and use it to prove the convergence of the algorithm. Paragraph 4 will deal with the integer case. The termination proof will result from a new uniform bound on the length of the nonparametric algorithm by the same techniques as those of the continuous case. The conclusion will review our results and point to some unsolved problems.

# 2 The dual simplex method

In this paper, **bold** letters will denote vectors with integer or rational coefficients. The notation $\mathbf{x} \geq 0$, where $\mathbf{x}$ is $n$-dimensional, will mean:

$$\forall i \in [1, n] : \mathbf{x}_i \geq 0.$$

Given an $m \times n$ matrix $M$ and an $m$-dimensional vector $\mathbf{v}$ , our aim is to solve the following problem:

Let $\mathsf{F}$ be the set:

$$\mathsf{F} = \{\mathbf{x} | \mathbf{x} \geq 0, M\mathbf{x} + \mathbf{v} \geq 0\}. \tag{1}$$

Decide whether $\mathsf{F}$ is empty, and, if not, select one element of $\mathsf{F}$ according to some preference criterium.

$\mathsf{F}$ is the set of feasible solutions. In operation research, it is usual to use a linear preference function:

$$\mathbf{x} \text{ is preferable to } \mathbf{y} \equiv \mathbf{c}.\mathbf{x} < \mathbf{c}.\mathbf{y},$$

where $\mathbf{c}$ is an $n$-dimensional vector and . denote the scalar product. This relation, however, is not an order. This lead to difficulties known in the litterature as degeneracy problems. For reasons which have been given above, we will rank the points of $\mathsf{F}$ according to the lexicographic order, noted as $\ll$ in the sequel. There would be no difficulty to extend our theory to the linear case on condition that $\mathbf{c} \geq 0$.

The problem will be solved by a succession of changes of variables, until we find ourselves in a situation where the solution is “obvious”. A linear change of variables is specified by an $n \times n$ matrix $P$ and an $n$-vector $\mathbf{u}$; the old variables $\mathbf{x}$ are given in term of the new ones, $\mathbf{y}$, by:

$$\mathbf{x} = P\mathbf{y} + \mathbf{u}, \tag{2}$$

and the new feasible set is:

$$\mathsf{F}' = \{P\mathbf{y} + \mathbf{u} | P\mathbf{y} + \mathbf{u} \geq 0, MP\mathbf{y} + M\mathbf{u} + \mathbf{v} \geq 0\}. \tag{3}$$

A common generalization of (1) and (3) is:

$$\mathsf{F} = \{A\mathbf{y} + \mathbf{b} | \mathbf{x} = A\mathbf{y} + \mathbf{b} \geq 0, \mathbf{z} = C\mathbf{y} + \mathbf{d} \geq 0, \mathbf{y} \geq 0\}. \tag{4}$$

Initially, $A$ is a unit matrix, $\mathbf{b}$ is null, $C$ is $M$ and $\mathbf{d}$ is $\mathbf{v}$. We may consider $A$ and $B$ as two blocks of an $(m+n) \times n$ matrix $S$, $\mathbf{b}$ and $\mathbf{c}$ as an $m+n$ vector $\mathbf{t}$ and $\mathbf{x}$ and $\mathbf{z}$ as an $m+n$ vector $\mathbf{w}$. In the course of the resolution process, vector $\mathbf{w}$ will stay fixed. The unknown vector $\mathbf{y}$ initially is a subset of $\mathbf{w}$ (namely $\mathbf{x}$) and will stays so, but the selection will change as the solution progress. In mathematical programming terminology, $[S\mathbf{t}]$ is the problem tableau; the $\mathbf{y}$'s are the basic variables; the variables of $\mathbf{w}$ which do not belong to $\mathbf{y}$ are non-basic.

$S_{\bullet j}$ (resp. $S_{i\bullet}$) will be the $j$-th column vector (resp. the $i$-th row vector) of $S$. The change of variable (2) is legitimate, first, if there is a value of $\mathbf{y}$ associated to each value of $\mathbf{x}$, i.e., if $P$ is invertible. Second, $\mathbf{y} \geq 0$ must be a consequence of the fact that $\mathbf{x}$ belongs to $\mathsf{F}$.

What are the "obvious" cases? Suppose first that there is a row $i$ of $S$ such that $S_{i\bullet} \leq 0$ and $\mathbf{t}_i < 0$. Then, since $\mathbf{x} \geq 0$, there is no possible way of having

$$S_{i\bullet}.\mathbf{x} + \mathbf{t}_i \geq 0.$$

In this case, $\mathsf{F}$ is empty.

Next, note that the initial $S$ is such that all its column vectors are lexico-positive (they begin by a string of zero followed by a one). Suppose we are able to maintain this property and that we reach a stage where $\mathbf{b} \geq 0$. Then there is a member of $\mathsf{F}$ associated to $\mathbf{x} = 0$, namely $\mathbf{b}$. Any increase in the value of $\mathbf{x}$ will add to $\mathbf{b}$ a lexico-positive vector; hence, $\mathbf{b}$ is the lexical minimum of $\mathsf{F}$.

This leads to the following technique for the construction of $P$. Select an index $i$ such that $\mathbf{t}_i < 0$, and a $j$ such that $S_{ij} > 0$. The corresponding row is:

$$\mathbf{w}_i = S_{\bullet i}.\mathbf{y} + \mathbf{t}_i. \tag{5}$$

Eliminate $\mathbf{y}_j$ in favor of $\mathbf{w}_i$ This is obviously an invertible transformation, and $\mathbf{w}_i \geq 0$ is guaranteed. $\mathbf{x}_j$ is given by:

$$\mathbf{x}_j = \mathbf{w}_i/S_{ij} - \sum_{k \neq j}(S_{ik}/S_{ij})\mathbf{y}_k - \mathbf{t}_i/S_{ij}. \tag{6}$$

After this transformation, the new tableau $[S'\mathbf{t}']$ will have as its column vectors:

$$\begin{aligned}
S'_{\bullet j} &= (1/S_{ij})S_{\bullet j}, && (7)\\
S'_{\bullet k} &= S_{\bullet k} - (S_{ik}/S_{ij})S_{\bullet j},\ k \neq j && (8)\\
\mathbf{t}' &= \mathbf{t} - (\mathbf{t}_i/S_{ij})S_{\bullet j} && (9)
\end{aligned}$$

Since $S_{ij}$ is positive, $S'_{\bullet j}$ will remain lexicopositive. For $S'_{\bullet k}$ to remain lexicopositive, $j$ must be choosen by the familiar rule: select the lexico-minimal caracteristic vector $S_{\bullet j}/S_{ij}$ from those with $S_{ij}$ positive. Element $S_{ij}$ is known as the pivot, and formulas (7) to (9) define a pivoting step. Note that since $\mathbf{t}_i$ is negative, $\mathbf{t}'$ will increase in the process.

In mathematical programming terminology, one says that variable $\mathbf{w}_i$ enters the basis and that $\mathbf{y}_j$ leaves it. The whole process may be seen as the selection of a submatrix $T$ of $S$ and the computation of its inverse. $T$ is the product of elementary matrices of the form:

$$\begin{pmatrix}
1 & 0 & \dots & & \\
0 & 1 & \dots & & \\
\dots & & & & \\
S_{i1} & \dots & S_{ij} & \dots & \\
\dots & & & & \\
 & \dots & 0 & 1 &
\end{pmatrix}$$

whose determinant is $S_{ij}$. Hence $D$, the determinant of $B$ is the product of the pivots. By Cramer's rule, we know that the elements of the transformed tableau will be fractions whose denominator is $D$ or one of its factors.

It is obvious that either we are in one of the two immediate solution cases, or else a choice of $i$ and $j$ is possible. Hence the algorithm does not stop unless the problem is solved. But the algorithm is nothing more than the selection of $n$ rows from the $(m+n)$ rows of $S$. There are only $C^n_{m+n}$ different choices, and since cycling is impossible by (9), the algorithm must terminate eventually. Note that the above bound does not depend on the particular value of $S$ or $\mathbf{t}$ but only on the dimensions of the problem, $m$ and $n$.

In practical terms, all we need for an implementation of the above algorithm is to record the tableau of the problem, i.e. the matrix $S$ and the vector $\mathbf{t}$. If we are given a linear preference function with positive coefficients, we just add it as the first line of the tableau, whose column vectors remain lexico-positive. This is the familiar dual simplex algorithm. One may observe that initially $n$ lines of S constitute a unit matrix, and that the pivoting steps simply scatter these lines in the problem tableau. It is customary not to record the unit part of S, thus reducing the complexity of a pivoting step from $O(n(m+n))$ to $O(mn)$. When working with lexicographic ordering, this optimization will disturb the numbering of the unknowns and hence change the final solution. In the interest of legibility, we will suppose in the sequel that we always work with the complete tableau; in a practical implementation, a more sophisticated programming technique must be used.

# 3 Continuous parametric programming

The next step in the solution is to suppose that the constant terms in (1) are no longer numbers but depend linearly on $p$ parameters. As a matter of convenience, we will suppose that these parameters (which are noted as a $p$-dimensional vector $\mathbf{z}$), are positive integers. The current version of (4) is then:

> Let $\mathsf{F}(\mathbf{z})$ be the set:
>
> $$\mathsf{F}(\mathbf{z}) = \{A\mathbf{x}+A'\mathbf{z}+\mathbf{b} | A\mathbf{x}+A'\mathbf{z}+\mathbf{b} \geq 0, C\mathbf{x}+C'\mathbf{z}+\mathbf{d} \geq 0, \mathbf{x} \geq 0\}. \quad (10)$$
>
> Decide for which values of $\mathbf{z}$, $\mathsf{F}(\mathbf{z})$ is empty. For other values of $\mathbf{z}$, express the lexico-minimal element of $\mathsf{F}(\mathbf{z})$ as a function of $\mathbf{z}$.

The idea of the solution method is to execute the dual simplex algorithm in a symbolic way, as one would do if working with pen and paper. For the algorithm to become a program, one needs to know the algebraic nature of each datum in the process. From an inspection of (10), it is clear that initially the elements of $S$ are numbers, while the elements of vector $\mathbf{t}$ are linear forms. Furthermore, inspection of (7) to (9) shows that this property remains true after a pivoting step, i.e. that the parameters remain confined in the constant terms. From (9), for instance, we deduce that the formula for a component of $\mathbf{t}'$ is:

$$\mathbf{t}'_k = \mathbf{t}_k - \mathbf{t}_i(S_{kj}/S_{ij}).$$

Here $\mathbf{t}_k$ and $\mathbf{t}_i$ are both linear forms, while $S_{kj}/S_{ij}$ is a number.

However, before a pivoting step, one must choose the pivot. Here again, as soon as $i$ is known, the choice of $j$ depends solely on $S$, and hence is independent of the value of $\mathbf{z}$. The choice of $i$, on the other hand, is controlled by the rule that $\mathbf{t}_i$ should be negative. This clearly depend on $\mathbf{z}$, and the only possibility is to split the problem in two subproblems according to the sign of $\mathbf{t}_i$. When this is done, the value of $\mathbf{z}$ is no longer arbitrary: in one subproblem it is constrained by $\mathbf{t}_i(\mathbf{z}) \geq 0$, and by the opposite inequality in the other one. When the next choice must be made, according to the sign of $\mathbf{t}_k$ (for instance), $\mathbf{t}_k(\mathbf{z}) \geq 0$ may or may not be compatible with $\mathbf{t}_i(\mathbf{z}) \geq 0$. If compatible, the value of $\mathbf{z}$ will be further constrained both by $\mathbf{t}_i(\mathbf{z}) \geq 0$ and $\mathbf{t}_k(\mathbf{z}) \geq 0$.

We are thus driven to introduce a further element in problem (10) : a set of linear constraints on the parameters,

$$K\mathbf{z} + \mathbf{h} \geq 0.$$

These inequalities on $\mathbf{z}$ will be called the context of the problem. Restricting $\mathbf{z}$ to positive integer values will simplify the handling of these constraints. We will suppose that the initial context of the problem is not unfeasible.

The algorithm will proceed by building a problem tree, i.e. a tree whose nodes are labelled by a problem tableau $\langle S, \mathbf{t}, K, \mathbf{h}\rangle$. In such a problem, the sign of component $\mathbf{t}_i$ of $\mathbf{t}$ may be positive, negative or unknown. It is unknown if both $\mathbf{t}_i(\mathbf{z}) \geq 0$ and $\mathbf{t}_i(\mathbf{z}) < 0$ are compatible with the context. The sign is known if only one of these inequalities is compatible with the context. Lastly, it will never be the case that none is compatible with the context: that would imply that the context is empty.

If all $\mathbf{t}_i$ are positive, then the node is a success leaf. If there is at least one negative $\mathbf{t}_i$, then we attempt a pivoting step according to (7-9), which may lead to failure or to success. In the first case, the node is a failure leaf. Its context delimits a region of the parameter space where $\mathsf{F}(\mathbf{z})$ is empty. In case of success, the original node will have an only son whose problem will be the result of the pivoting step.

In the remaining case, select a $\mathbf{t}_i$ whose sign is unknown. The original node will have two sons with the same problem tableau. In one of them, the context will be augmented by $\mathbf{t}_i(\mathbf{z}) \geq 0$, and in the other one by $\mathbf{t}_i(\mathbf{z}) < 0$.

It remains to say how the compatibility of a set of linear inequalities is to be tested. We have supposed that all numbers that enter in our algorithms are rationals. As a consequence, the context may be stored as a set of forms with integer coefficients. It is easy to bring $\mathbf{t}_i(\mathbf{z})$ to this form by multiplication by a suitable number; $\mathbf{t}_i(\mathbf{z}) < 0$ is brought to the canonical form $f(\mathbf{z}) \geq 0$ by changing all signs and subtracting one from the constant term. One is then left with the problem of deciding the feasability of a system of linear inequalities in integers. This is a nonparametric programming problem, which may be solved by well known techniques ([Gre71]); see also the following paragraph.

The resulting algorithm may be summarized in the following terms:

Algorithm Q

To solve the parametric continuous problem with tableau $\langle S, \mathbf{t}(\mathbf{z})\rangle$ in the context $K\mathbf{z} + \mathbf{h} \geq 0$ :

1. Determine the signs of the components of $\mathbf{t}(\mathbf{z})$ in the context $K\mathbf{z} + \mathbf{h} \geq 0$;
2. If all $\mathbf{t}_i(\mathbf{z})$ are positive, the solution is given by the first $|\mathbf{x}|$ components of $\mathbf{t}(\mathbf{z})$;
3. If there is a negative $\mathbf{t}_i(\mathbf{z})$, then either:
   (a) All elements of $S_{\bullet i}$ are negative, and the solution may be written as $\infty$, indicating that it does not exist;

(b) There is at least a positive $S_{ij}$; a pivoting step is executed, giving a new problem $\langle S', \mathbf{t}'(\mathbf{z})\rangle$. The solution of the initial problem is the same as that of the problem $\langle S', \mathbf{t}'(\mathbf{z})\rangle$ in the context $K\mathbf{z} + \mathbf{h} \geq 0$;

4. In the remaining case, select a $\mathbf{t}_i(\mathbf{z})$ whose sign is unknown; let $\mathbf{x}_+$ and $\mathbf{x}_-$ be respectively the solutions of $\langle S, \mathbf{t}(\mathbf{z})\rangle$ in the contexts:

$$\{K\mathbf{z} + \mathbf{h} \geq 0, \mathbf{t}_i(\mathbf{z}) \geq 0\},$$

$$\{K\mathbf{z} + \mathbf{h} \geq 0, \mathbf{t}_i(\mathbf{z}) < 0\}.$$

The solution of the initial problem is:

$$\mathbf{if}\ \mathbf{t}_i(\mathbf{z}) \geq 0\ \mathbf{then}\ \mathbf{x}_+\ \mathbf{else}\ \mathbf{x}_-.$$

## 3.1 The correctness proof

That the above algorithm is partially correct is obvious, since it does nothing but reproduce in a symbolic way the moves of the dual simplex algorithm. Does it always terminate?

Note first that the problem tree is finitely branching. A node has at most two sons (in case (4) above). Hence, by König lemma, if the tree is infinite it has an infinite branch. Second, the number of splitting steps between two pivoting steps is bounded by $m$, since there are only $m + n$ components of $\mathbf{t}(\mathbf{z})$ and since $n$ of those are always null. Last, note that by construction, all contexts are non void.

Select a node on the infinite branch whose distance to the root is greater than $mC^n_{m+n}$, and a value of $\mathbf{z}$ which belongs to its context. Executing the dual simplex algorithm for this value of $\mathbf{z}$ will lead to the choosen node in more than $C^n_{m+n}$ pivoting steps, in contradiction to a previously obtained bound.

## 3.2 An example

To bring the initial problem in the canonical form (10), one has to introduce new unknowns:

$$i' = m - i,$$

$$j' = n - j,$$

and to replace one equation by two opposite inequalities. The result is:

Find the lexical minimum of:

$$\mathsf{F}'(k,m,n) = \{(i',j') \mid \begin{array}{l} 0 \le i' \le m, \\ 0 \le j' \le n, \\ -2i' - j' - k + 2m + n \ge 0, \\ 2i' + j' + k - 2m - n \ge 0\}. \end{array}$$

The initial tableau is:

| A | $i'$ | $j'$ | 1 | $k$ | $m$ | $n$ | Sign |
|---|---|---|---|---|---|---|---|
| $i'$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $j'$ | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| $a$ | -1 | 0 | 0 | 0 | 1 | 0 | + |
| $b$ | 0 | -1 | 0 | 0 | 0 | 1 | + |
| $c$ | -2 | -1 | 0 | -1 | 2 | 1 | ? |
| $d$ | 2 | 1 | 0 | 1 | -2 | -1 | ? |

The first four rows are null or positive, while the last two rows sign is unknown. We must split the program in two subprograms according to the sign of $(-k + 2m + n)$.

Suppose first this linear form is non-negative. This clearly implies that the last row cannot be positive. (This fact is easily proved by showing that the corresponding program is unfeasible; we omit the details for brevity sake). A pivoting step is indicated; the variable $d$ will enter the basis in place of variable $j'$. The resulting tableau is:

| B | $i'$ | $d$ | 1 | $k$ | $m$ | $n$ | Sign |
|---|---|---|---|---|---|---|---|
| $i'$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $j'$ | -2 | 1 | 0 | -1 | 2 | 1 | + |
| $a$ | -1 | 0 | 0 | 0 | 1 | 0 | + |
| $b$ | 2 | -1 | 0 | 1 | -2 | 0 | ? |
| $c$ | 0 | -1 | 0 | 0 | 0 | 0 | 0 |
| $d$ | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

Context: $-k + 2m + n \ge 0$

Here, all rows have non negative constant terms with the exception of the $b$ row. There are two cases according to the sign of $k - 2m$.

If $k - 2m \ge 0$, all constant terms are non negative and the solution is apparent:

$$i' = 0,$$

$$j' = -k + 2m + n.$$

If not, another pivoting step is necessary: $b$ enters the basis in place of $i'$.

| C | $b$ | $d$ | 1 | $k$ | $m$ | $n$ | Sign |
|---|---|---|---|---|---|---|---|
| $i'$ | 1/2 | 1/2 | 0 | -1/2 | 1 | 0 | + |
| $j'$ | -1 | 0 | 0 | 0 | 0 | 1 | + |
| $a$ | -1/2 | -1/2 | 0 | 1/2 | 0 | 0 | + |
| $b$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $c$ | 0 | -1 | 0 | 0 | 0 | 0 | 0 |
| $d$ | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

$$\begin{aligned}\text{Context}: \ & k - 2m \geq 0 \\ & -k + 2m + n \geq 0\end{aligned}$$

Here, all constant terms are non negative, and the solution is:

$$\begin{aligned} i' &= -k/2 + m, \\ j' &= n. \end{aligned}$$

The remaining case is $-k + 2m + n < 0$. Going back to the first tableau, we see that all coefficients in the $c$ row are negative and hence that the program is unfeasible.

We may splice all the above results and express the resulting formula in term of the original unknowns $i$ and $j$:

$$\begin{pmatrix} i \\ j \end{pmatrix} = \begin{array}{ll} \textbf{if} & (-k + 2m + n \geq 0) \\ \textbf{then} & \textbf{if } (k - 2m \geq 0) \\ & \textbf{then} \begin{pmatrix} m \\ k - 2m \end{pmatrix} \\ & \textbf{else} \begin{pmatrix} k/2 \\ 0 \end{pmatrix} \\ \textbf{else } \infty. & \end{array}$$

Since the problem is two dimensional, the result could have been obtained by inspection of figure 1. The interest of our method is that it can be used whatever the dimensionality of the problem.

# 4 The Integer Case

We must now take into account the restriction of $\mathbf{x}$ to integer values. There are several techniques, for which the reader is referred to [Gre71], [Tah75] or [Min83]. In the present context, we must select an algorithm whose moves may be carried out even if the constant terms depend linearly on integer parameters, and whose complexity is uniformly bounded with respect to the constant terms. The cutting plane algorithm of [Gom63] answers to these requirements. Paragraph 4.1 describes it; the convergence proof is given in paragraph 4.2. In the next paragraph we will devise its symbolic version; the termination proof will follow in a straightforward way.

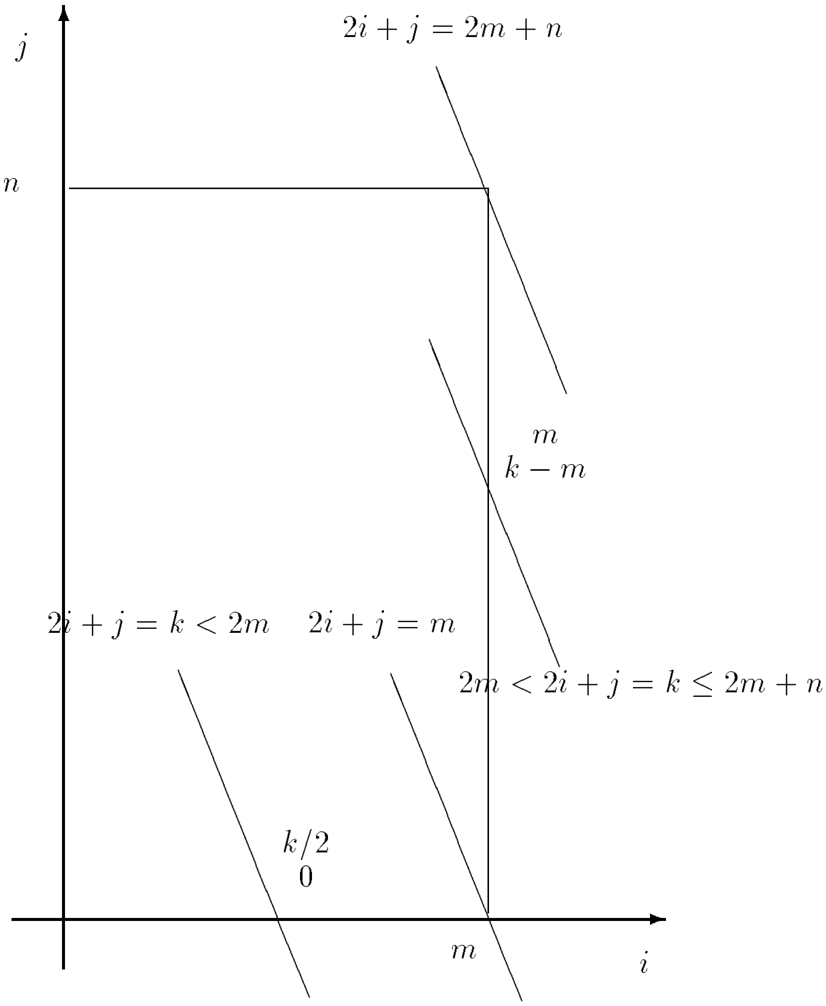


Figure 1: A simple parametric programming problem

## 4.1 An integer programming algorithm

The problem to be solved may be expressed in the following way:

> Let $\mathsf{F}$ be the set:
>
> $$\mathsf{F} = \{\mathbf{x} | \mathbf{x} \geq 0, M\mathbf{x} + \mathbf{v} \geq 0, \mathbf{x} \in \mathsf{N}\} \tag{11}$$
>
> where $\mathsf{N}$ is the set of positive integers. Decide whether $\mathsf{F}$ is empty and, if not, select its lexical minimum.

We will suppose, with no loss of generality, that $M$ and $\mathbf{v}$ have integer coefficients, which implies that $M\mathbf{x}+\mathbf{v}$ is an integer vector. The solution will proceed, as in 2, by a succession of variable changes according to (5) and (6). As we have seen, the new independant variable, $\mathbf{y}_j$, is either one of the $\mathbf{x}$ or one of the constraints. Hence $\mathbf{y}_j$ will also be constrained to integer values. However, as (6) shows, not all integer values of $\mathbf{y}_j$ will result in integral values for $\mathbf{x}$. Hence an integrity constraint for $\mathbf{x}$ must be introduced explicitly. The correct generalization both of (4) and (11) is:

$$\mathsf{F} = \{A\mathbf{x} + \mathbf{b} | A\mathbf{x} + \mathbf{b} \in \mathsf{N}, C\mathbf{x} + \mathbf{d} \in \mathsf{N}, \mathbf{x} \in \mathsf{N}\}. \tag{12}$$

The dual simplex algorithm as given by (7) to (9) will eventually terminate, with non negative vectors $\mathbf{b}$ and $\mathbf{d}$. There is no guarantee that these vectors are integers; it follows that the solution is not necessarily given by $\mathbf{x} = 0$. The only information we have is that the solution $\mathbf{u}$ is in $\mathsf{F}$, that the column vectors of $A$ are lexico-positives and that, since $\mathbf{x} \geq 0$,

$$\mathbf{b} \ll \mathbf{u}.$$

The principle of the cutting plane method [Gom63] is to add a new constraint to (12) in such a way as to exclude the continuous optimum while keeping all feasible integer points. The new constraint or cut must be a consequence of:

$$A\mathbf{x} + \mathbf{b} \in \mathsf{N},$$

$$\mathbf{x} \in \mathsf{N}.$$

To derive a cut, select the first row $i$ of $A$ such that $\mathbf{b}_i$ is not an integer. If there is no such row, the current $\mathbf{b}$ is integral and the program is solved. Let $D$ be the common denominator of the $A_{ij}$ and of $\mathbf{b}_i$. If:

$$\sum_j S_{ij}\mathbf{x}_j + \mathbf{t}_i \in \mathsf{N},$$

then

$$\sum_j (DS_{ij})\mathbf{x}_j + (D\mathbf{t}_i) \equiv 0 \pmod D. \tag{13}$$

It is interesting to reduce this congruence to lowest terms by replacing all integers by their remainder when divided by $D$. Let us use the sign % (in the C fashion) for the remaindering operator:

If $a = bq + r$ where $0 \le r < b$, then $a\%b = r$.

(13) is equivalent to:

$$\sum_j ((DS_{ij})\%D)\mathbf{x}_j \equiv (-D\mathbf{t}_i)\%D \pmod D, \tag{14}$$

or

$$\sum_j ((DS_{ij})\%D)\mathbf{x}_j = (-D\mathbf{t}_i)\%D + kD. \tag{15}$$

Since the left hand side is positive, while $(-D\mathbf{t}_i)\%D$ is positive and less than $D$, we see that $k$ is non negative and hence that:

$$\sum_j ((DS_{ij})\%D)\mathbf{x}_j - (-D\mathbf{t}_i)\%D \ge 0. \tag{16}$$

Note also that:

$$\sum_j \frac{(DS_{ij})\%D}{D}\mathbf{x}_j - \frac{(-D\mathbf{t}_i)\%D}{D} = k, \tag{17}$$

a positive integer. Hence we may add as a cut:

$$\sum_j \frac{(DS_{ij})\%D}{D}\mathbf{x}_j - \frac{(-D\mathbf{t}_i)\%D}{D} \in \mathsf{N}, \tag{18}$$

and the format of the problem will not be changed.

The new row will have a negative constant term; to restore feasibility, one or more pivoting steps must be executed. The algorithm will proceed until either the feasible set is proved to be empty or a feasible integer solution is found. Since cuts are consequence of the program constraints, adding a cut does not eliminate any integer solution; if the feasible set is found to be empty, this prove that the initial program had no integer solution. On the contrary, if a solution is found, an argument similar to the one given in 2 will show that it is the lexico-minimal one.

## 4.2 The convergence proof

The classical convergence proof (see e.g. [Gre71] or [Sch86]) is based on the observation that the constant term $\mathbf{b}$ in program (12) lexicographically increase at each step of the algorithm, but is bounded by an eventual solution. Since we are equally interested in cases where there is no solution, our first step will be to construct an enlarged program whose solution always exists and is simply related to the solution, if any, of the original program. The convergence proof will then follow along classical lines. Finally, with the help of a theorem of Cook et al. [WHAE86], we will give a uniform bound on the maximum number of cuts.

### 4.2.1 The enlarged problem

Starting from program (11), let $x_0$ be a new unknown; let $\mathsf{F}_+$ be the program:

$$\mathsf{F}_+ = \{x_0, \mathbf{x} | x_0, \mathbf{x} \in \mathsf{N}; x_0 + M\mathbf{x} + \mathbf{v} \in \mathsf{N}\}.$$

It is quite clear that $\mathsf{F}_+$ is not empty; take

$$x_0 = \max(0, -\mathbf{v}_i)$$

and

$$\mathbf{x} = 0.$$

Next, if $\mathsf{F}$ is not empty and has lexical minimum $\mathbf{u}$, then $\langle 0, \mathbf{u}\rangle$ is the lexical minimum of $\mathsf{F}_+$. Conversely, if $\langle u_0, \mathbf{u}\rangle$ is the lexical minimum of $\mathsf{F}_+$, then either $u_0 = 0$, and $\mathbf{u}$ is the minimum of $\mathsf{F}$, or $u_0 > 0$, and $\mathsf{F}$ has no solution.

In the course of the resolution of $\mathsf{F}_+$, as long as the new variable $x_0$ remains in the basis, all columns of the tableau except the first will start with at least one zero. Hence, the first column will never be choosen as the pivot column unless it is the only candidate, wich means that $x_0$ leaves the basis, that its optimum value will be non zero and that $\mathsf{F}$ is unfeasible. In other words, at any given step in the resolution of $\mathsf{F}$, the tableau is obtained from the corresponding tableau for $\mathsf{F}_+$ by deleting the first column and the first row. It is easily seen that this property is also true when cuts are added, since source rows are the same and so is $D$. We conclude, then, that the complexity of $\mathsf{F}_+$ is an upper bound on the complexity of $\mathsf{F}$. Hence it is sufficient to give a convergence proof in the case where an integral solution exits.

### 4.2.2 The convergence proof again

Let

$$\mathsf{F} = \{\mathbf{x} | M\mathbf{x} + \mathbf{v} \in \mathsf{N}; \mathbf{x} \in \mathsf{N}\} \tag{19}$$

be a program with solution $\mathbf{u}$. Let $\mathsf{F}_n$ be the transformed program just after the $n$-th cut:

$$\mathsf{F}_n = \{A^{(n)}\mathbf{y} + \mathbf{b}^{(n)} | A^{(n)}\mathbf{y} + \mathbf{b}^{(n)} \in N; C^{(n)}\mathbf{y} + \mathbf{d}^{(n)} \in \mathsf{N}\}$$

and let $\mathsf{F}'_n$ (with similar notations) be the program just after the pivoting step on the $n$-th cut. Let $\sigma(n)$ be the source row for the $n$-th cut. By (18) the constant term in the $n$-th cut is:

$$-\frac{(-D^{(n)}\mathbf{b}^{(n)}_{\sigma(n)})\% D^{(n)}}{D^{(n)}}$$

while the pivot is:

$$\frac{(D^{(n)}A^{(n)}_{\sigma(n)j})\% D^{(n)}}{D^{(n)}}.$$

By (9),

$$\mathbf{b}'^{(n)}_{\sigma(n)} = \mathbf{b}^{(n)}_{\sigma(n)} + \frac{(-D^{(n)}\mathbf{b}^{(n)}_{\sigma(n)})\%D^{(n)}}{D^{(n)}} \frac{D^{(n)}}{(D^{(n)}A^{(n)}_{\sigma(n)j})\%D^{(n)}} A^{(n)}_{\sigma(n)j},$$

where $\mathbf{b}^{(n)}_{\sigma(n)}$ and $A^{(n)}_{\sigma(n)j}$ are both positive. We note the upper bound:

$$(D^{(n)}A^{(n)}_{\sigma(n)j})\%D^{(n)} \leq D^{(n)}A^{(n)}_{\sigma(n)j},$$

from which follows:

$$\mathbf{b}'^{(n)}_{\sigma(n)} \geq \frac{D^{(n)}\mathbf{b}^{(n)}_{\sigma(n)} + (-D^{(n)}\mathbf{b}^{(n)}_{\sigma(n)})\%D^{(n)}}{D^{(n)}}. \tag{20}$$

Let

$$\mathbf{b}^{(n)}_{\sigma(n)} = q(n) - \frac{r(n)}{D(n)} \tag{21}$$

where $q(n)$ and $r(n)$ are both integers with:

$$0 \leq r(n) < D(n).$$

From the above follows:

$$\mathbf{b}'^{(n)}_{\sigma(n)} \geq q(n) > \mathbf{b}^{(n)}_{\sigma(n)}. \tag{22}$$

The last inequality is strict since, for the row $\sigma(n)$ to be choosen as the source row, $\mathbf{b}^{(n)}_{\sigma(n)}$ must be fractional. The algorithm is such that $\mathbf{b}$ is lexicographically increasing, which implies that $\mathbf{b}_1$ is non decreasing in the usual sense. We have just proved that each time the first row is used as the source row, there is another integer (namely $\{\mathbf{b}^{(n)}_1\}$) between $\mathbf{b}^{(n)}_1$ and $\mathbf{b}'^{(n)}_1$.

Consider now a cut whose source row is not row 1, which implies that $\mathbf{b}_1$ is integral. Let $j$ be the pivot column of the next step. Then either $S_{1j} = 0$, and $\mathbf{b}_1$ does not change, or $S_{1j} > 0$, and $\mathbf{b}_1$ increases. If the resulting value is integral, then $\mathbf{b}_1$ has increased by at least 1; if not, the next cut will use row 1, and $\mathbf{b}_1$ will increase beyond $\{\mathbf{b}_1\}$.

Let us say that a cut is a 1-cut if either row 1 is the source row, or $S_{1j} > 0$ where $j$ is the index of the pivot column in the next step. From the above discussion, we see that if the $n$-th cut is a 1-cut, then there is an integer in $[\mathbf{b}^{(n)}_1, \mathbf{b}^{(n+1)}_1]$, and these integers form a strictly increasing sequence. If $K_n$ is the number of 1-cuts between steps 1 and $n$, then:

$$\mathbf{b}^{(n)}_1 - \mathbf{b}^{(0)}_1 \geq K_n - 1. \tag{23}$$

We know that $\mathsf{F}$ has a lexical minimum $\mathbf{u}$, that $\mathbf{u}$ belongs to $\mathsf{F}_n$ at each step of the process, and hence that there exists values of $\mathbf{y}$ such that:

$$\mathbf{u} = A^{(n)}\mathbf{y} + \mathbf{b}^{(n)},\ \mathbf{y} \geq 0. \tag{24}$$

From this we deduce, since $A^{(n)}$ columns are lexico-positive, that:

$$\mathbf{b}^{(n)} \ll \mathbf{u}$$

and specifically that:

$$\mathbf{b}_1^{(n)} \leq \mathbf{u}_1.$$

This in turn implies that the total number of 1-cuts is bounded. There is a step $N_1$ such that $\mathbf{b}_1^{(n)} = \mathbf{b}_1^{(N_1)}$ for $n \geq N_1$; $\mathbf{b}_1^{(N_1)}$ is integral. By the definition of an 1-cut, after step $N_1$, pivoting is confined to those columns whose first element is null. In any row $i$, let $J_i^+$ (resp. $J_i^0$, $J_i^-$) be the set of column indices $j$ such that $S_{ij} > 0$ (resp. $S_{ij} = 0$, $S_{ij} < 0$). From (24) we deduce the bounds:

$$\forall j \in J_1^+ : 0 \leq \mathbf{y}_j \leq \frac{\mathbf{u}_1 - \mathbf{b}_1^{(n)}}{A_{1j}^{(n)}} \tag{25}$$

which stays valid for the rest of the procedure, since the first row of the tableau does not change anymore. In row 2, any $A_{2j}^{(n)}$ with $j$ in $J_1^0$ will be non negative, to insure that the corresponding column is lexico-positive. In other words:

$$J_2^- \subseteq J_1^+, \tag{26}$$

from which follows the bound:

$$\mathbf{b}_2^{(n)} = \mathbf{u}_2 - \sum_j A_{2j}^{(n)}\mathbf{y}_j \leq \mathbf{u}_2 + \sum_{j \in J_2^-} (-A_{2j}^{(n)}) \frac{\mathbf{u}_1 - \mathbf{b}_1^{(n)}}{A_j^{(n)}}. \tag{27}$$

From this we deduce, by the same argument as above, that the number of cuts on the second row is bounded. The same argument may be repeated for all rows of the tableau. It follows that, after a finite number of steps, all coordinates of $\mathbf{b}$ will be integral, and the algorithm will terminate.

In the sequel, we will say that row $i$ has *settled* after step $N_i$ if $\mathbf{b}_i^{(n)} = \mathbf{b}_i^{(N_i)}$ for all $n \geq N_i$. We have just proved that $\mathbf{b}_1^{(N_1)} = \mathbf{a}_1$ and that after step $N_1$, for $j \in J_1^+$, $\mathbf{y}_j = 0$. This is tantamount to saying that, as soon as row 1 has settled, the remaining unknowns are found by solving a deflated program, whose tableau is constructed from the current $S$ by deleting the first row and all columns in $J_1^+$.

### 4.2.3 A uniform bound

From (23) we deduce that the total number of 1-cuts is less than:

$$K = \{\mathbf{u}_1 - \mathbf{b}_1^{(0)} + 1\}, \tag{28}$$

where $\mathbf{b}^{(0)}$ is the continuous optimum. This number may be uniformly bounded by a technique adapted from a result of Cook, Gerards, Schrijver and Tardos [WHAE86] (see also [Sch86, Theorem 17.2]).

$\mathbf{u}$ and $\mathbf{b}^{(0)}$ are both in the original feasible set $\mathsf{F}$:

$$M\mathbf{u} + \mathbf{v} \geq 0, \mathbf{u} \geq 0,$$

$$M\mathbf{b}^{(0)} + \mathbf{v} \geq 0, \mathbf{b}^{(0)} \geq 0.$$

These constraints may be summarized as:

$$S\mathbf{u} + \mathbf{t} \geq 0,$$

$$S\mathbf{b}^{(0)} + \mathbf{t} \geq 0,$$

where S is the matrix $\begin{pmatrix} I \\ M \end{pmatrix}$ and $\mathbf{t}$ is the vector $\begin{pmatrix} 0 \\ v \end{pmatrix}$. Distribute the rows of $S$ in two matrices $S_+$ and $S_-$ with the properties that:

$$S_+\mathbf{u} > S_+\mathbf{b}^{(0)},$$

$$S_-\mathbf{u} \leq S_-\mathbf{b}^{(0)}.$$

Let $t$ be distributed accordingly into vectors $\mathbf{t}_+$ and $\mathbf{t}_-$. Consider the cone

$$\mathsf{C} = \{\mathbf{x} | S_+\mathbf{x} \geq 0; S_-\mathbf{x} \leq 0\}. \tag{29}$$

It is clear that $\mathbf{u} - \mathbf{b}^{(0)} \in \mathsf{C}$. $\mathsf{C}$ is generated by a set of linearly independent integer vectors $\{\mathbf{a}_1, ..., \mathbf{a}_t\}$ whose coordinates are no larger than $D$, where $D$ is the largest $n \times n$ subdeterminant from $S$. Hence:

$$\mathbf{u} - \mathbf{b}^{(0)} = \sum_k \alpha_k \mathbf{a}_k,\ \alpha_k \geq 0,$$

and by Caratheodory's theorem, there are at most $n$ non-zero terms in the above sum.

It is easy to prove that all $\mathbf{u}'$ of the form:

$$\mathbf{u}' = \mathbf{b}^{(0)} + \sum_k \alpha'_k \mathbf{a}_k,$$

where $0 \leq \alpha'_k \leq \alpha_k$, are in $\mathsf{F}$. From this we deduce that for all $k$, $\mathbf{b}^{(0)} + \alpha_k \mathbf{a}_k$ is in $\mathsf{F}$. Since $\mathbf{b}^{(0)}$ is the lexicographic minimum of $\mathsf{F}$, this implies that either $\alpha_k = 0$ (in which case $\mathbf{a}_k$ may be ignored in the sequel), or $0 \ll \mathbf{a}_k$.

We claim that $\alpha_k \leq 1$. If this were not true, we could construct $\mathbf{u}' = \mathbf{u} - \mathbf{a}_k$; $\mathbf{u}'$ is in $\mathsf{F}$ since $0 \leq \alpha'_k = \alpha_k - 1 \leq \alpha_k$, and $\mathbf{u}'$ is integral. Furthermore, $\mathbf{u}' \ll \mathbf{u}$, which contradict the definition of $\mathbf{u}$.

From this we deduce:

$$K \leq nD. \tag{30}$$

If the original problem is not integer feasible, $K$ is bounded by the number of cuts for program $\mathsf{F}_+$, which is easily seen to be less than $(n+1)nD$ (since a subdeterminant for $S_+$ may be written as an alternate sum of $(n+1)$ subdeterminants from $S$).

The above bound is the uniform bound we require for the termination proof of the parametric version of Gomory's algorithm. It is known that the bound:

$$|\mathbf{u} - \mathbf{b}^{(0)}|_\infty \leq nD$$

is strict. Whether the above bound share the same property is unknown and is left for future research.

### 4.3 The parametric version of the Gomory algorithm

We already know how to parametrize the dual simplex; it remains only to show how to parametrize the construction of a cut. Refer back to formula (18). The first point is the determination of $D$. We noted in 2 that $D$ is a factor of the determinant of the basis, which is equal to the product of the pivots. It is a simple matter to keep track of this product. The construction of the cut is equally valid if the determinant is used in lieu of the common denominator.

Next, the $S_{ij}$ are known numbers; there is no difficulty in computing $(DS_{ij})\%D$. The problem lies in the evaluation of $(-D\mathbf{t}_i(\mathbf{z}))\%D$, a non linear function of $\mathbf{z}$. Let us introduce a new notation. If $\mathbf{t}$ is a linear form and $f$ is a numerical function, $(f\mathbf{t})$ will stand for the form whose coefficients are obtained from those of $\mathbf{t}$ by componentwise application of $f$. One has, for instance:

$$(-\mathbf{t})(\mathbf{z}) = -\mathbf{t}(\mathbf{z}),$$

but this commutativity property is not always true, as for instance in:

$$(\mathbf{t}\%D)(\mathbf{z}) \equiv \mathbf{t}(\mathbf{z})(mod D).$$

To obtain the required cut, introduce a new parameter

$$q = ((-D\mathbf{t}_i)\%D)(\mathbf{z}) \div D. \tag{31}$$

$q$ is a positive integer, since both the components of $\mathbf{z}$ and the coefficients of $((-D\mathbf{t}_i)\%D)$ are non negative. $q$ is completely defined by two linear constraints:

$$0 \leq ((-D\mathbf{t}_i)\%D) - qD \leq D - 1, \tag{32}$$

which must be added to the context. Obviously, a $q$ such that (32) is true always exists. Hence adding (32) to the context does not restrict the possible values of $\mathbf{z}$; it merely gives a linear definition of $q$.

The analogue of (15) is:

$$\sum_j ((DS_{ij})\%D)\mathbf{x}_j = ((-D\mathbf{t}_i)\%D)(\mathbf{z}) - qD + kD. \tag{33}$$

A cut follows in the usual fashion. The complete parametric integer programming algorithm is analogous to algorithm (Q) with the following changes:

ALGORITHM N

- In step (3b), keep track of $D$, the product of the pivots.
- Step (2) is replaced by the following. If all $\mathbf{t}_i(\mathbf{z})$ are positive, select the earliest row $i$ such that $(DS_{ij})\%D$ and $(D\mathbf{t}_i(\mathbf{z}))\%D$ are not identically 0. If no such row exists (in particular if $D = 1$), the solution has been found; it is given by the first $|\mathbf{x}|$ components of $\mathbf{t}(\mathbf{z})$.

  If such a row exists, add (32) to the context. Add to the tableau the new row:

$$\sum_j \frac{(DS_{ij})\%D}{D}\mathbf{x}_j - \sum_k \frac{T_{ik}}{D}\mathbf{z}_k - \frac{T_{i0}}{D} + q \geq 0, \tag{34}$$

  where the $T_{ik}$ are the coefficients of $((-D\mathbf{t}_i)\%D)$. Start again at step (1).

The convergence proof is a straightforward consequence of the uniform bound of paragraph 4.2.3. If the solution tree is infinite it has an infinite branch. Since there are exactly $n$ candidate rows for a cut, on the infinite branch there is a row which does not settle, and a node $\alpha$ such that no row settles beyond $\alpha$. The remainder of the branch address the solution of a deflated program which is constructed as indicated in 4.2.3. Let $r$ be the deflated unknown count, and let $D$ be the largest of all $r \times r$ subdeterminant in the deflated tableau. Select a node $\beta$ which lies more than $r(r+1)D$ 1-cuts away from $\alpha$, and a value of $\mathbf{z}$ which belongs to the context of $\beta$. It is clear that solving the deflated problem for this value of $\mathbf{z}$ will contradict (30).

## 4.4 The introductory exemple again

The beginning of the computation is the same as 4.3. The solution associated to tableau (B) clearly is integral; hence, (B) is a success node. In the case of (C), the solution is fractional and the determinant $D$ is 2. The source congruence is:

$$b + d - k + 2m \equiv 0 \pmod 2.$$

In the notation of (32), $\mathbf{t}_i$ is $-k + 2m$, and $((-D\mathbf{t}_i)\%D)$ is simply $k$. To construct a cut, we introduce the new parameter

$$q = k \div 2,$$

and the cut is:

$$e = b/2 + d/2 - k/2 + q \geq 0.$$

Two inequalities are added to the context:

$$0 \leq k - 2q \leq 1.$$

After a pivoting step on $e$ and $b$, one gets:

| D | $e$ | $d$ | 1 | $k$ | $m$ | $n$ | $q$ | Sign |
|---|---|---|---|---|---|---|---|---|
| $i'$ | 1 | 0 | 0 | 0 | 1 | 0 | -1 | + |
| $j'$ | -2 | 1 | 0 | -1 | 0 | 1 | 2 | ? |
| $a$ | -1 | 0 | 0 | 0 | 0 | 0 | 1 | + |
| $b$ | 2 | -1 | 0 | 1 | 0 | 0 | -2 | + |
| $c$ | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $d$ | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $e$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$$\begin{aligned}\text{Context}:\ & -1 - k + 2m \geq 0 \\ & -k + 2m + n \geq 0 \\ & k - 2q \geq 0 \\ & 1 - k + 2q \geq 0\end{aligned}$$

The new determinant is $2 \times 1/2 = 1$. The sign of the $j'$ row is unknown. In case $-k + n + 2q \geq 0$, the solution is:

$$i' = m - q,$$

$$j' = -k + n + 2q.$$

In the opposite case, we first execute a pivoting step on $j'$ and $d$, giving:

| E | $e$ | $j'$ | 1 | $k$ | $m$ | $n$ | $q$ | Sign |
|---|---|---|---|---|---|---|---|---|
| $i'$ | 1 | 0 | 0 | 0 | 1 | 0 | -1 | + |
| $j'$ | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $a$ | -1 | 0 | 0 | 0 | 0 | 0 | 0 | + |
| $b$ | 0 | -1 | 0 | 0 | 0 | 1 | 0 | + |
| $c$ | -2 | -1 | 0 | -1 | 0 | 1 | 2 | $-$ |
| $d$ | 2 | 1 | 0 | 1 | 0 | -1 | -2 | + |
| $e$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$$
\begin{aligned}
\text{Context}: \; & -1-k+2m \geq 0 \\
& -k+2m+n \geq 0 \\
& k-2q \geq 0 \\
& 1-k+2q \geq 0 \\
& -1+k-n-2q \geq 0
\end{aligned}
$$

Here, all rows are positive with the exception of $c$, whose constant term is -$k+n+2q$. But $-k+n+2q<0$ is in the context of (E), and hence row $c$ is negative. Since both coefficients ($-2$ and $-1$) are negative, (E) is a failure node. The algorithm has terminated.
We may write the final solution as:

$$
\begin{pmatrix} i \\ j \end{pmatrix} = \begin{array}{lll}
\textbf{if} & (-k+2m+n \geq 0) & \\
\textbf{then} & \textbf{if} & (k-2m \geq 0) \\
 & \textbf{then} & \begin{pmatrix} m \\ k-2m \end{pmatrix} \\
 & \textbf{else} & \textbf{if } (-k+n+2(k \div 2)) \geq 0) \\
 & & \textbf{then} \begin{pmatrix} k \div 2 \\ k-2(k \div 2) \end{pmatrix} \\
 & & \textbf{else } \infty \\
\textbf{else } \infty. & &
\end{array} \tag{35}
$$

From this the value of `a[k]` may be easily computed. An interesting fact is that we have detected another case in which `a[k]` is not defined : $n-(k-2(k \div 2)) < 0$. This occurs only for odd values of $k$ if $n=0$; it would be very easy to overlook this error.

# 5 Conclusion

The algorithm we have given has been implemented and has been found to be reliable for small problems as are found in the semantic analysis of computer programs. Its theoretical complexity is quite high; in practice, we have found it to share the well known property of the simplex, which while exponential in the worst case, has a high probability of being polynomial. In fact, we have found the complexity of the algorithm to be commensurate to the complexity of the solution, and one cannot ask for less.

The running time may be reduced by various devices. We note that part of the problem tableau is a unit matrix, which does not carry usefull information. The corresponding rows may be deleted, thus reducing the computational burden by a factor of $m/(n+m)$. This is the so-called revised form of the simplex algorithm. In our case, we must keep track of the deleted rows in order not to disturb the lexicographic ordering.

Experience shows that most of the running time is spent in testing the feasibility of auxilliary systems in step (1) of the algorithm. A large speed-up is obtained if we detect cases in which the sign of $\mathbf{t}_i(\mathbf{z})$ is "obvious"; this include:

- after a pivoting step, the constant term in the pivot row is null;
- the constant term of the new cut is always negative;
- if all coefficients of $\mathbf{t}_i(\mathbf{z})$ are of one and the same sign, then since $\mathbf{z} \geq 0$, $\mathbf{t}_i(\mathbf{z})$ is of this sign;
- in a pivoting step, we add to $\mathbf{t}_i(\mathbf{z})$ a positive multiple of the pivot column. If both addends are of the same sign, the sign of the result is not changed.

Since the termination of the algorithm depends on distinguishing between integers and non-integers, care must be taken to avoid rounding errors. It is possible to use infinite precision rational arithmetic as is available on some programming environments (e.g. `bc` in the Unix system or the rational arithmetic package of some versions of Lisp). This is, however, unduly wastefull. Note that at each steps $DS_{ij}$ and $D\mathbf{t}_i(\mathbf{z})$ (where $D$ is the determinant of the basis, i.e. the product of the pivots) are integral. The problem tableau may be represented by the triple $\langle D, DS, D\mathbf{t}(\mathbf{z})\rangle$, in which all elements are integers. The algorithm may be entirely reformulated in this new representation (in fact, the analogue of (7)-(9) are slightly simplified). Rounding errors disappear, to be replaced by potential overflows, a much simpler proposition.

While the algorithm is guaranteed to terminate with a correct solution, this solution is by no means unique. In step (2) and (3), and also in a cut construction, there are degrees of freedom, which may be exploited to speed up the algorithm (e.g. by selecting the "best" pivoting row or the "deepest" cut).

In our case, there is one more choice: the choice of the splitting row in step (4). For instance, if the $c$ and $d$ rows of our exemple are interchanged, the solution is:

$$\begin{pmatrix} i \\ j \end{pmatrix} = \begin{array}{llll} \textbf{if} & (k-2m-n \leq 0) & & \\ \textbf{then} & \textbf{if} & (-k+2m+n \geq 0) & \\ & \textbf{then} & \begin{pmatrix} 0 \\ 0 \end{pmatrix} \textbf{ else } \infty & \\ & \textbf{else} & \textbf{if} & (k-2m \geq 0) \\ & & \textbf{then} & \begin{pmatrix} m \\ k-2m \end{pmatrix} \\ & & \textbf{else} & \textbf{if } (-k+n+2(k \div 2)) \geq 0) \\ & & & \textbf{then } \begin{pmatrix} k \div 2 \\ k-2(k \div 2) \end{pmatrix} \\ & & & \textbf{else } \infty \\ \textbf{else } \infty. & & & \end{array} \tag{36}$$

This is equivalent to but slightly more complex than (35). We would be interested in using this degree of freedom to obtain the "simplest" solution. This however is a very difficult problem, which is left for future research.

# References


[Dan63] G. B. Dantzig. *Linear Programming and extensions.* Princeton University Press, Princeton, NJ, 1963.

[Gal79] T. Gal. *Postoptimal Analysis, Parametric programming and related topics.* Mac Graw Hill, 1979.

[Gom63] R. E. Gomory. An algorithm for integer solutions to linear programs. In R. L. Graves and P. Wolfe, editors, *Recent Advances in Math. Programming*, chapter 34, pages 269–302. Mac-Graw Hill, New York, 1963.

[Gre71] H. Greenberg. *Integer programming.* Academic Press, 1971.

[Min83] Michel Minoux. *Programmation Mathématique, théorie et algorithmes.* Dunod, Paris, 1983.

[Sch86] A. Schrijver. *Theory of linear and integer programming.* Wiley, NewYork, 1986.

[Tah75] H. Taha. *Integer Programming.* Academic Press, 1975.

[WHAE86] Cook W., Gerards A. M. H., Schrijver A., and Tardos E. Sensitivity theorems in integer linear programming. *Mathematical Programming*, 34:251–264, 1986.

```
@ARTICLE{Feau:88b,
        AUTHOR = "Paul Feautrier", VOLUME = 22, PAGES = "243--268",
        TITLE ="Parametric Integer Programming",
        JOURNAL = "RAIRO Recherche Op\'{e}rationnelle",
        MONTH = Sep, YEAR = 1988}
```