SIAM J. COMPUT.
Vol. 23, No. 6, pp. 1179–1192, December 1994


004

# SIMPLE AND FAST ALGORITHMS FOR LINEAR AND INTEGER PROGRAMS WITH TWO VARIABLES PER INEQUALITY*

DORIT S. HOCHBAUM† AND JOSEPH (SEFFI) NAOR‡

**Abstract.** The authors present an $O(mn^2 \log m)$ algorithm for solving feasibility in linear programs with up to two variables per inequality which is derived directly from the Fourier–Motzkin elimination method. (The number of variables and inequalities are denoted by $n$ and $m$, respectively.) The running time of the algorithm dominates that of the best known algorithm for the problem, and is far simpler. Integer programming on monotone inequalities, i.e., inequalities where the coefficients are of opposite sign, is then considered. This problem includes as a special case the simultaneous approximation of a rational vector with specified accuracy, which is known to be NP-complete. However, it is shown that both a feasible solution and an optimal solution with respect to an arbitrary objective function can be computed in pseudo-polynomial time.

**Key words.** linear programming, integer programming

**AMS subject classifications.** 05C85, 68Q25, 90C05, 90C10, 90C27

## 1. Introduction.

In this paper we examine linear and integer programming problems with two variables per inequality. The problem of computing a feasible solution in the linear (or fractional) case has been investigated extensively. Shostak [18] suggested that a linear program with two variables per inequality can be represented as a graph: since each inequality contains two variables, one can represent the linear program by a graph which has a vertex for each variable, and an additional vertex $x_0$. Any inequality involving two variables is represented as an edge between the respective pair of vertices. As for inequalities involving only one variable (upper and lower bounds on variables), these are represented as edges to and from vertex $x_0$. We denote the number of variables by $n$, and the number of inequalities by $m$. (W.l.o.g. we can assume that $m \geq n$.) The graph consists therefore of $n + 1$ vertices and $m$ edges, and there may be multiple edges between any pair of vertices.

Shostak [18] proved that feasibility can be tested by following paths and cycles in this graph, and thus laid the foundation for all subsequently considered algorithms for the problem. This feasibility test was refined to a polynomial algorithm by Aspvall and Shiloach [1], and later still to an $O(mn^3 \log m)$ strongly polynomial algorithm by Megiddo [14]. Recently, Cohen and Megiddo [4] obtained new algorithms for this problem: (i) they presented a new $O(mn^2(\log m + \log^2 n))$ time algorithm; (ii) they also gave a randomized algorithm for finding a feasible solution in the special case of monotone inequalities (to be defined later) with an expected running time of $O(n^3 \log n + mn \log^3 m \log n + mn \log^5 n)$. (This randomized algorithm was later generalized to hold for the non-monotone case as well in [2]; however, it follows from [5] that the non-monotone case can be reduced to the monotone case at no extra cost.) The main feature common to all of these algorithms is determining upper and lower bounds for each variable by following paths and cycles in the graph.

The first result we present is an $O(mn^2 \log m)$ algorithm for the feasibility problem. This algorithm is faster (although only for $m < n^{o(\log n)}$), and moreover it is *simpler* than all other known algorithms. The backbone of our algorithm is the Fourier–Motzkin elimination method

*Received by the editors July 1, 1993; accepted for publication July 9, 1993. A preliminary version of this paper appeared in the Proceedings of the 2nd "Integer Programming and Combinatorial Optimization" Conference (IPCO), CMU, 1992, pp. 44-59.

†Department of Industrial Engineering and Operations Research, University of California, Berkeley, California 94720 (dorit@hochbaum.berkeley.edu). This research was supported in part by Office of Naval Research grant ONR N00014-91-J-1241.

‡Department of Computer Science, Technion, Haifa 3200, Israel (naor@cs.technion.ac.il). Most of this work was done while the author was at the Computer Science Department, Stanford University and supported by contract ONR N00014-88-K-0166. The author was also supported in part by contract ONR N00014-91-J-1241.

(Introduced by Fourier (1827), and discovered later by Dines (1918–1919) and Motzkin (1936); see [17] for details.) In general, this algorithm does not run in polynomial time because it may generate an exponential number of inequalities in the process of eliminating variables. However, we show how to implement this algorithm *efficiently* for linear programs where each inequality may contain at most two variables. First, at each elimination step, the number of inequalities on every edge adjacent to the variable currently to be eliminated is reduced to two. This serves to control the exponential growth of the number of inequalities. In addition, we maintain the inequalities corresponding to two variables as upper and lower envelopes, where the envelopes (which are piecewise linear functions) are characterized by their breakpoints. This representation allows us to dispose of redundant inequalities in each elimination step quickly by examining all breakpoints associated with the variable currently to be eliminated.

The analogue of the Fourier–Motzkin procedure in computational logic is *resolution*. Using resolution, one obtains a satisfying assignment to a set of clauses (in, say, propositional logic) by eliminating the variables one by one. It is known that resolution can be efficiently implemented for the case of 2-SAT clauses, i.e., the satisfiability problem, where each clause may contain at most two literals. This follows since every elimination step generates 2-SAT clauses, and the total number of 2-SAT clauses is always bounded by a polynomial. Our algorithm may be viewed as an efficient implementation of resolution for the case of linear constraints with two variables per inequality.

A linear program with two variables per inequality is called *monotone* if each inequality is of the form $ax_i - bx_j \leq c$, where both $a$ and $b$ are positive. We will consider integer programming problems on monotone inequalities. We note that the aforementioned reduction from the non-monotone case to the monotone case does not preserve integrality.

Lagarias [11] has shown that the problem of deciding whether a given rational vector $\alpha$ has a simultaneous approximation of specified accuracy with respect to the maximum norm, with denominator $Q$ in a given interval $1 \leq Q \leq N$, is NP-complete. The problem of deciding the feasibility of a monotone system in integers is a generalized form of this question and hence NP-complete as well (it is obviously in NP).

The set of feasible solutions of a monotone system can be shown to form a *distributive lattice* where the *join* and *meet* operations are defined to be maximum component-wise and minimum component-wise, respectively. This has been observed before by Veinott [20]. We present an algorithm that computes the solution vectors corresponding to the top and bottom of the lattice. The lattice structure is crucial for the algorithm, and the manner in which the search for a feasible solution is conducted guarantees that if one exists, then we are going to find the solution which is at the top (or bottom). The running time of this algorithm is a polynomial which depends on the sum of the number of integer valued points in each one-dimensional projection of the feasible polytope in the fractional case. Hence, this algorithm is pseudo-polynomial in the case when the variables in the integer program are bounded. Also, in this case the problem is weakly NP-complete.

It is interesting to note that the strongly polynomial feasibility algorithm for linear inequalities with two variables per inequality does not extend to a strongly polynomial *optimization* algorithm over such inequalities. (It is only known that when the objective function consists of $d$ variables, then there is a strongly polynomial algorithm when $d$ is fixed, i.e., it is exponential in $d$.)

In contrast, for the integer case, we present a pseudo-polynomial algorithm for the optimization problem over a monotone system with an arbitrarily long objective function (that is, with up to $n$ variables in the objective). We note that the optimization problem over a non-monotone system is NP-complete in the strong sense, since the vertex cover problem is a special case of it. The algorithm hinges on the following two observations:

- The elements of a distributive lattice can be represented as closed subsets of a directed graph (of pseudo-polynomial size) which is derived from the lattice.
- A linear objective function defines a modular function on the lattice which in turn implies that the lattice element of optimal cost corresponds to the closed subset of optimal cost (when costs are properly defined).

The complexity of computing the closed subset of optimal cost is bounded by a polynomial in the size of the graph, i.e., it can be computed in pseudo-polynomial time. Even though the directed graph that represents the lattice is of pseudo-polynomial size, it has a succinct description, i.e., it can be encoded in polynomial space. This provides a compact encoding of the complete feasible solution set of a monotone system of inequalities.

Finally, we present an application of our Fourier–Motzkin algorithm to identifying *fat* polytopes. Fat polytopes are those containing a sphere which circumscribes a unit hypercube, and hence must contain an integer point. For polytopes derived from inequalities with two variables per inequality, the procedure for identifying fat polytopes runs in strongly polynomial time, and thereby can be viewed as an efficient heuristic for finding a feasible integer solution. A strongly polynomial algorithm for a related problem of finding the largest sphere contained in a polytope is presented as well.

**2. Efficient implementation of the Fourier–Motzkin algorithm.** In this section we show how the Fourier–Motzkin elimination method for finding a feasible solution of a linear program can be implemented efficiently when the number of variables in each inequality is at most two. We begin by an informal description of the method for a general linear program. (The reader is referred to [17, pp. 155–156] for more details.)

Let the variables of the linear program be $x_1, \ldots, x_n$ and let the set of inequalities be denoted by $E$. The variables are eliminated one by one. At step $i$, the linear program will only contain variables $x_i, \ldots, x_n$; the set of inequalities at step $i$ is denoted by $E_i$, where initially $E_1 = E$. To eliminate variable $x_i$, all the inequalities in which $x_i$ participates are partitioned into two sets, $L$ and $H$. The set $L$ contains all the inequalities which are of the form $x_i \geq l$, and the set $H$ contains all the inequalities which are of the form $x_i \leq h$, where $l$ and $h$ are linear functions. To obtain the set $E_{i+1}$, for all $l \in L$ and $h \in H$, a new inequality $l \leq h$ is added to $E_i$, and all the inequalities in $L$ and $H$ are eliminated from it. The number of new inequalities produced is $|H| \cdot |L|$. The next theorem is immediate.

THEOREM 2.1. *The linear program $E_{i+1}$ has a feasible solution if and only if the linear program $E_i$ has a feasible solution.*

Hence, a feasible solution can be computed recursively for $E_{i+1}$ and then extended to $E_i$. The main drawback of this method is that the running time is not necessarily polynomial, i.e., in general the number of inequalities may grow exponentially.

The discussion henceforth is restricted to inequalities that contain at most two variables per inequality. It is interesting to note that Nelson [15] proved that when implementing the Fourier–Motzkin method in this case, the total number of inequalities is bounded by $m \cdot n^{\log n}$.

As mentioned in the introduction, an equivalent representation of the linear program is by the graph $G = (V, E)$. The vertex set $V$ contains vertices $x_0, x_1, \ldots, x_n$; an edge between vertex $x_i$ and $x_j$ (for $1 \leq i, j \leq n$) represents the set of inequalities in which $x_i$ and $x_j$ participate. The vertex $x_0$ is needed to represent inequalities that contain precisely one variable, i.e., an edge from $x_i$ to $x_0$ denotes an inequality of the form $x_i \leq a$ or $x_i \geq a$ for some constant $a$.

The main feature that allows for the efficient implementation of the Fourier–Motzkin algorithm is the following. The set of inequalities that correspond to an edge between $x_i$ and $x_j$ is represented in the $(x_i, x_j)$ plane as two envelopes, an upper envelope and a lower envelope. The feasible region of $x_i$ and $x_j$ is in between the two envelopes and it is not hard to

see that each envelope is a *piecewise linear* function that can be represented by its breakpoints (see Fig. 1(a)).

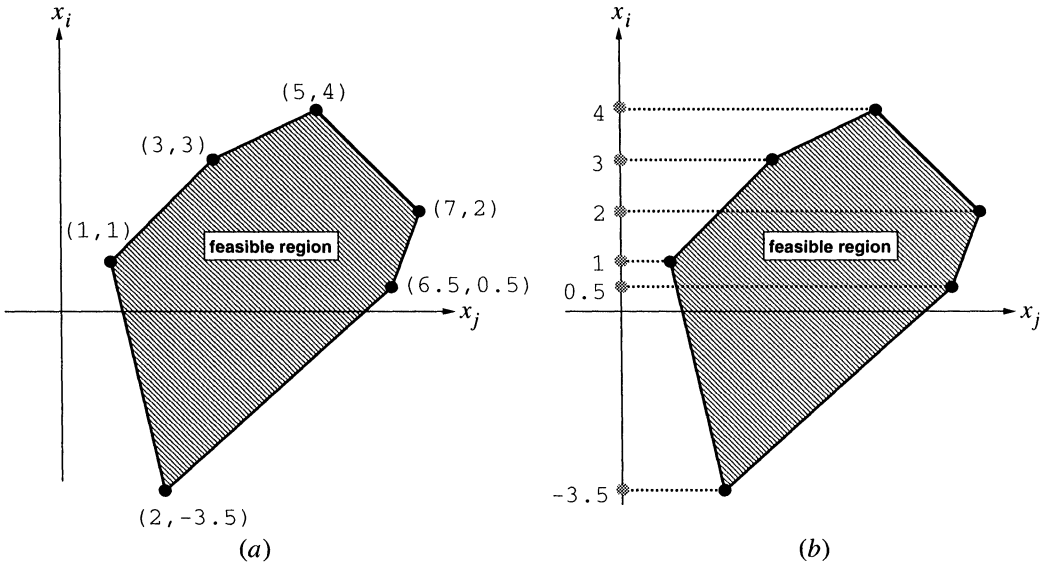


FIG. 1. (a) *The feasible region defined by the inequalities containing $x_i$ and $x_j$ is a piecewise linear function defined by its breakpoints.* (b) *The set $B_j = \{-3.5, 0.5, 1, 2, 3, 4\}$ is the set of breakpoints projected on the $x_i$ axis.*

The following procedure of Aspvall and Shiloach [1] plays a crucial role in our algorithm. This procedure was used by [14] and [4] as well. Let $x_i^{\min}$ and $x_i^{\max}$ denote the respective minimum and maximum feasible values of $x_i$. That is, any value assigned to $x_i$ from the range $[x_i^{\min}, x_i^{\max}]$ can be complemented to a feasible solution. If the feasible range of some of the variables is unbounded, then there exist numbers bounded by a polynomial in the binary representation of the data [17], such that if $x_i^{\min}$ and $x_i^{\max}$ are set to them, the existence of a feasible solution is assured.

PROCEDURE 2.1 [1]. *Given a variable $x_i$ and a value $\lambda$, it can be decided in $O(mn)$ operations whether (i) $\lambda < x_i^{\min}$, (ii) $\lambda > x_i^{\max}$, or (iii) $x_i^{\min} \leq \lambda \leq x_i^{\max}$.*

The main idea underlying Procedure 2.1 is propagating the implications of the equality $x_i = \lambda$ in a manner very similar to the Bellman–Ford algorithm for computing all shortest paths from a single source. We remark that even if the linear program in hand is infeasible, the procedure may still provide one of the above three answers. In this case, infeasibility will be detected by our algorithm at a later stage. (Note for example the case in which the linear system consists of two independent subsystems, one feasible and one infeasible.)

We are now ready to provide a high-level view of the algorithm. The main idea is that the number of inequalities in which $x_i$ (the variable to be eliminated) participates can be significantly reduced using Procedure 2.1. It should be mentioned that a similar idea was used by Megiddo [14] to obtain upper and lower bounds on the feasible values of variables. The following is performed at step $i$ of the Fourier–Motzkin algorithm. Let $G_i$ denote the graph corresponding to the linear program $E_i$.

1. Let the neighbors of $x_i$ in the graph $G_i$ be $x_{i_1}, \ldots, x_{i_d}$.
   Let $B_j$ $(1 \leq j \leq d)$ denote the set of breakpoints of the edge $(x_i, x_{i_j})$ projected on the $x_i$ coordinate (see Fig. 1(b)).

2. Merge the $d$ sorted sequences $B_i$ into a sorted sequence $B$. (Let the sorted sequence be $b_1, \ldots, b_k$.)
3. Perform a binary search on the sequence $B$. The aim of the search is to obtain either
   (a) a breakpoint $b_l \in B$ such that $x_i^{\min} \leq b_l \leq x_i^{\max}$, or
   (b) an interval $[b_l, b_{l+1}]$ $(1 \leq l < k)$ such that $b_l < x_i^{\min}$ and $x_i^{\max} < b_{l+1}$.
4. In step 3a, variable $x_i$ is assigned the value $b_l$ and "contracted" with vertex $x_0$ in graph $G_i$.
   In step 3b, the number of inequalities on each edge adjacent to $x_i$ is reduced to at most two (see Fig. 2). Now, the generic Fourier–Motzkin elimination step is applied to variable $x_i$.

Let us further elaborate on how the algorithm is implemented and analyze its complexity. The following invariant is maintained throughout the algorithm; we defer its proof to the end of the discussion. It is obviously true initially.

INVARIANT 2.1. *The number of breakpoints on an edge is at most* $O(m)$.

By the invariant, the cardinality of the set $B$ is at most $O(mn)$. The binary search at step 3 is performed by successive calls to Procedure 2.1. At each call, either a breakpoint which is feasible for $x_i$ is discovered, or the number of breakpoints to be examined is reduced by half. Hence, the complexity of sorting the set $B$ and performing the binary search is at most $O(mn \log m)$. We should remark that in the course of the elimination process, to bound the running time of Procedure 2.1 by $O(mn)$, we run it on the original graph $G$ and not on the current graph $G_i$. However, $G$ is updated as follows. For each eliminated variable (say $x$) that was assigned a value (say $a$) at step 3a, two inequalities are added to graph $G$: $x \leq a$ and $x \geq a$. If $x$ is already connected to $x_0$, then the respective bounds are updated according to the most restrictive bound, or an inconsistency is detected and the algorithm terminates with a discovery that the system is infeasible.

In step 3a, the linear program $E_{i+1}$ is obtained from $E_i$ by assigning the value $b_l$ to the variable $x_i$. Otherwise, in step 3b, the generic Fourier–Motzkin elimination step is applied. Notice that the number of inequalities on each edge adjacent to $x_i$ is reduced to at most two (see Fig. 2). (We assume that the intersections of the upper and lower envelopes (up to two) are also counted among the original breakpoints.) In addition, two more inequalities, $b_l \leq x_i$ and $x_i \leq b_{l+1}$, are added to the linear program $E_i$.

Let $x_{i_p}$ and $x_{i_q}$ be any two variables that are adjacent to $x_i$. The edge $(x_i, x_{i_q})$ and the edge $(x_i, x_{i_q})$ may each contain at most two inequalities; hence, at step 4, the Fourier–Motzkin elimination step adds up to four new inequalities between the variables $x_{i_p}$ and $x_{i_q}$. These four inequalities are added to the set of inequalities that already exist between them. The running time of adding a new inequality to an already existing envelope is $O(\log m)$ time. This follows since the existing set of inequalities includes at most $O(m)$ inequalities (Invariant 2.1), which is represented as an (upper and lower) envelope, i.e., as a sorted sequence of breakpoints. Adding a new inequality amounts to identifying where to insert the newly created breakpoint in the existing sequence, which can be done using a binary search. We note that it may be the case that, as a result of adding a new breakpoint, many other breakpoints can disappear. Since there are at most $\binom{n}{2}$ pairs of neighbors, the complexity of this step is at most $O(n^2 \log m)$.

To prove Invariant 2.1, notice that for each variable that is eliminated, the number of breakpoints added to an edge is a constant, and hence the invariant is maintained. In fact, the number of breakpoints on an edge will never exceed $m + 4n$ throughout the execution of the algorithm.

At the end of the elimination step, we are left with two variables, $x_0$ and $x_n$. We now backtrack and assign values to the inequalities as follows. Choose any feasible value in the feasible range for $x_n$. Now choose a feasible value for $x_{n-1}$, that satisfies the inequalities w.r.t.

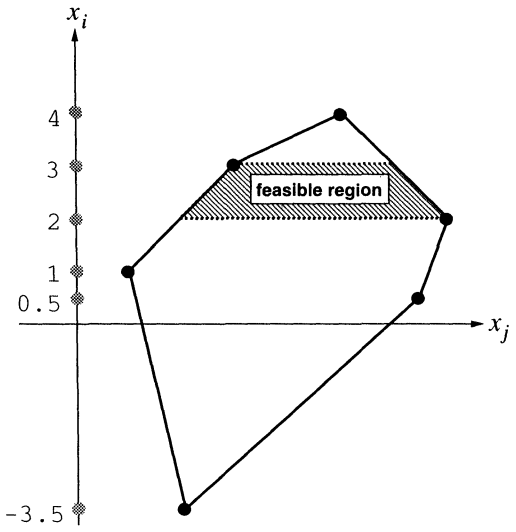


FIG. 2. *Step* 3b *of the algorithm: for example,* $b_l = 2$ *and* $b_{l+1} = 3$. *Consequently, the number of inequalities involving* $x_i$ *and* $x_j$ *is reduced to two. In addition, there are two more inequalities:* $2 \preceq x_i$ *and* $x_i \preceq 3$.

$x_n$ and $x_0$. Continue inductively by determining a value for $x_i$, based on the inequalities of $x_i$ and $x_j$ for $j > i$, and the range determined by the upper and lower bound inequalities with $x_0$. Since there are now at most two inequalities on each edge, the running time for the entire backtracking process is $O(n^2)$.

The correctness of the next theorem follows from the above discussion.

THEOREM 2.2. *The complexity of eliminating a variable in the algorithm is* $O(mn \log m)$. *Hence, the complexity of the entire algorithm is* $O(mn^2 \log m)$.

**3. Integer programming on monotone inequalities.** A linear program with two variables per inequality is called *monotone* if for every inequality, the coefficients of the two variables have opposite signs. We begin by studying the properties of the set of feasible vectors in the case of monotone inequalities. This set can be looked upon as a partial order under the following definition of dominance. Given two feasible vectors, $L_1$ and $L_2$, we say that $L_1 \succeq L_2$ if for all components $i$, $L_1(i) \geq L_2(i)$. Let $\mathcal{L}$ denote the set of all feasible vectors of a monotone system. We prove that $\mathcal{L}$ has the nice property that it forms a *distributive lattice*. This property will turn out to be very useful for finding a feasible solution and optimizing with respect to an objective function. It was previously observed by Veinott [20]. A distributive lattice is a partial order in which

1. Each pair of elements has a greatest lower bound, or *meet*, denoted by $a \wedge b$, so that $a \wedge b \preceq a$, $a \wedge b \preceq b$, and there is no element $c$ such that $c \preceq a$, $c \preceq b$ and $a \wedge b \prec c$.
2. Each pair of elements has a least upper bound, or *join*, denoted by $a \vee b$, so that $a \preceq a \vee b$, $b \preceq a \vee b$, and there is no element $c$ such that $a \preceq c$, $b \preceq c$ and $c \prec a \vee b$.
3. The *distributive* laws hold, namely $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ and $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$.

To prove that $\mathcal{L}$ is, in fact, a distributive lattice, we define appropriately the *meet* and *join* operations. The meet of two vectors $L_1$ and $L_2$ is defined to be the vector where each component is the minimum of the two corresponding components in $L_1$ and $L_2$. The join of two vectors is defined similarly where minimum is replaced by maximum. (See Fig. 3 for an example.)

THEOREM 3.1 [20]. *The partial order $(\mathcal{L}, \preceq)$ of feasible vectors forms a distributive lattice under the above definitions of meet and join.*

*Proof.* We first establish that $(\mathcal{L}, \preceq)$ is a lattice. To do that, we prove one case, other cases follow similarly. Let $L_1 = (u_1, \ldots, u_n)$, $L_2 = (v_1, \ldots, v_n)$, and let $L = L_1 \vee L_2 = (w_1, \ldots, w_n)$. We show that $L$ is also a feasible solution vector. For a particular inequality $ax_i - bx_j \leq c$, we know that

$$au_i - bu_j \leq c; \quad av_i - bv_j \leq c.$$

If, for example, $w_i = u_i$ and $w_j = v_j$, then since $b$ is positive, $bu_j \leq bv_j$, and the inequality holds for solution vector $L$ as well.

Let $a$, $b$, and $c$ be any integers. Then, $\min(a, \max(b, c)) = \max(\min(a, b), \min(a, c))$ and $\max(a, \min(b, c)) = \min(\max(a, b), \max(a, c))$. Hence, the distributive laws hold for the lattice $\mathcal{L}$. □

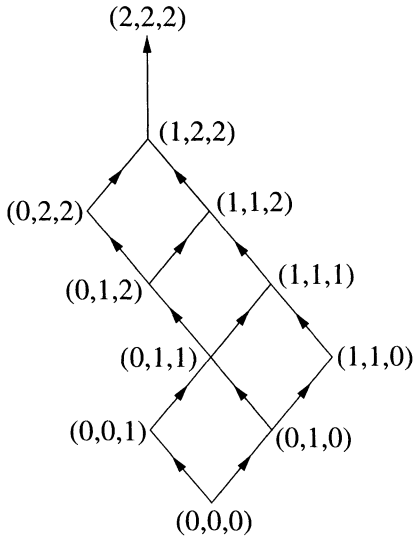


FIG. 3. *The sublattice of integral solutions of a monotone system. Each solution vector is of the form* $(x, y, z)$.

Notice that the lattice property holds in both the fractional and the integer case, and in fact the set of integer feasible solutions is a *sublattice* of the lattice of feasible solutions. From now on $\mathcal{L}$ will denote the lattice in the integer case and we restrict the discussion to this lattice. It is easy to see that the lattice properties imply that a lattice has a unique minimum and maximum, denoted by $B$ (bottom) and $T$ (top), respectively.

The problem of checking whether an integer monotone system has a feasible solution was shown to be NP-complete by Lagarias [11]. This was shown by proving that the following problem, *good simultaneous approximation*, is NP-complete. An instance of this problem consists of a vector of rationals, $\alpha = (a_1/b_1, \ldots, a_n/b_n)$, and positive integers $N$, $s_1$, and $s_2$. The question is whether there exists an integer $Q$, $1 \leq Q \leq N$, such that

$$\max_{1 \leq i \leq n} \{Qa_i/b_i\} \leq \frac{s_1}{s_2},$$

where $\{\beta\}$ denotes the distance of $\beta$ to the closest integer. This problem can be expreseed as an instance of finding an integer feasible solution $(x_1, \ldots, x_n, Q)$ for the following monotone system:

$$-s_1 \cdot B \le s_2 \left( \frac{a_i \cdot B}{b_i} \cdot Q - B \cdot x_i \right) \le s_1 \cdot B, \qquad 1 \le i \le n,$$

$$\text{where} \quad B = b_1 \cdot b_2 \cdots b_n \qquad \text{and} \qquad 1 \le Q \le N.$$

However, we will show in §§3.1 and 3.2 that for the case of bounded variables, both the feasibility problem and the optimality problem with respect to an arbitrary objective function can be solved in pseudo-polynomial time over a monotone system of inequalities. Consequently, integer programming over monotone inequalities with bounded variables is only weakly NP-complete.

**3.1. Integer feasibility over monotone inequalities.** In this section we show how a feasible integer solution can be found. More specifically, our aim is to compute the feasible solution which corresponds to the top of the lattice, i.e., the feasible vector whose components are maximal. The same procedure with a slight modification can be applied to find the solution corresponding to the bottom of the lattice.

During the course of this procedure a current solution vector $L = (x_1, \ldots, x_n)$ (which is infeasible) is maintained with the invariant that $L \succeq T$. The initial value of $L$ is the top of the fractional lattice where each component is rounded downward to the nearest integer.

It should be noted that in the monotone case, the top (or bottom) of the lattice in the fractional case can be computed via a simple modification of the algorithm defined in §2. In step 3a of the algorithm, instead of assigning a value to the variable that is eliminated, two consecutive breakpoints are computed with the following property. One breakpoint belongs to the feasible region, and the other breakpoint (the larger one) does not belong to it. The algorithm continues similarly to step 3b. In the backtracking process, the highest feasible value is chosen for each variable.

The generic step in the algorithm is as follows. Traverse all the inequalities in the linear program in some arbitrary order. If an inequality $ax_i - bx_j \le c$ is invalid, then it is *validated* by updating the value of $x_i$ as follows:

$$x_i \leftarrow \left\lfloor \frac{c + bx_j}{a} \right\rfloor.$$

The algorithm terminates if either all inequalities traversed in a single step are valid, or $L \preceq B$.

THEOREM 3.2. *The algorithm tests in time* $O(mn^2 \log m + m \cdot \sum_{i=1}^{n} (x_i^{\max} - x_i^{\min} + 1))$ *whether a monotone system of inequalities has a feasible solution.*

*Proof.* Assume that the given system of inequalities has a feasible solution. We show that the invariant that $L \succeq T$ is maintained throughout the algorithm. Given an invalid inequality $ax_i - bx_j \le c$, where $x_i = u_i$ and $x_j = u_j$, it is validated by decreasing $x_i$. Assume that the values of $x_i$ and $x_j$ in $T$ are $\hat{x}_i$ and $\hat{x}_j$, respectively. By the invariant, $u_j \ge \hat{x}_j$. Hence,

$$\hat{x}_i \le \left\lfloor \frac{c + b\hat{x}_j}{a} \right\rfloor \le \left\lfloor \frac{c + bu_j}{a} \right\rfloor = u_i$$

and the invariant is maintained.

It follows from the invariant that we will never need to backtrack, and since the value of a variable is always decreased by at least one unit whenever an inequality is validated, the running time of the algorithm is bounded by $O(mn^2 \log m + m \cdot \sum_{i=1}^{n} (x_i^{\max} - x_i^{\min} + 1))$. This running time is pseudo-polynomial since the feasible range of the variables is bounded.

If the monotone system does not have an integer feasible solution, this will be detected when $L \preceq B$, since $T \succeq B$ must hold. □

Recently, T. Feder (private communication) observed that an integer feasible solution can be computed in pseudo-polynomial time in the non-monotone case when the variables are bounded.

**3.2. Integer optimization over monotone inequalities.** In this section we consider the following integer optimization problem:

$$\min \quad \sum_{i=1}^{n} w_i \cdot x_i$$

$$\text{subject to} \quad a_k x_i - b_k x_j \leq c_k, \quad k = 1, \ldots, m,$$

$$1 \leq i, j \leq n \quad \text{and} \quad a_k,\ b_k \geq 0.$$

We show that the optimal solution can be computed in pseudo-polynomial time where the polynomial depends on $\sum_{i=1}^{n}(x_i^{\max} - x_i^{\min})$. Recall that the optimization problem over a non-monotone system is NP-complete in the strong sense since vertex cover is a special case of it.

DEFINITION 3.3. *Let $f$ be a function defined on a lattice $\mathcal{L}$, and let $a, b \in \mathcal{L}$. The function $f$ is called modular if $f(a) + f(b) = f(a \vee b) + f(a \wedge b)$.*

It is straightforward to verify that any linear objective function defined on an integer monotone system is modular.

DEFINITION 3.4. *For a directed graph $G$, a subset $S$ is said to be closed if for every $s \in S$, all its predecessors, i.e., all vertices $s'$ for which there exists a directed path from $s'$ to $s$, belong to the subset $S$.*

We first review our scheme for minimizing with respect to an objective function. The following theorem in lattice theory (see [6, p. 72, Thm. 9] and [7, Thm. 2.2.1]) is relevant to our result.

THEOREM 3.5. *Given a distributive lattice $\mathcal{L}$, a partial order can always be associated with it, such that a one-to-one correspondence can be established between its closed subsets and the elements of $\mathcal{L}$.*

The proof of this theorem is constructive and it implies an algorithm for constructing the partial order. In general, there may be more than one partial order that has the above property; we denote by $I(\mathcal{L})$ the partial order obtained by following the proof of Theorem 3.5 and call it the *generic partial order*.

Suppose now that a modular function $f$ is defined on the lattice $\mathcal{L}$. It can be shown that in this case, the elements of the partial order can be assigned costs in such a way that the lattice element of optimal cost would correspond to the closed subset of $I(\mathcal{L})$ of optimal cost. Computing a closed subset of optimal cost in a partial order is a well-known problem and its complexity is bounded by a polynomial in the size of $I(\mathcal{L})$ [16]. (The size of $I(\mathcal{L})$ is pseudo-polynomial in the case of our lattice.)

The disadvantage of computing with the generic partial order $I(\mathcal{L})$ is that its structure is rather complicated, and it seems that it cannot be described compactly, i.e., in polynomial space (as opposed to pseudo-polynomial space). Instead, we present a directed graph, denoted by $G(\mathcal{L})$, that also has the property that a one-to-one correspondence exists between its closed subsets and the elements of $\mathcal{L}$. The advantage of this graph is that it can be encoded in polynomial space via an algorithm which has a short (polynomial) description.

The rest of the section is organized as follows. In §3.2.1 we define the directed graph $G(\mathcal{L})$ and prove that it has the desired properties. In §3.2.2 we show how to compute a closed subset of minimum cost in $G(\mathcal{L})$. For the sake of completeness, we discuss in §3.2.3 how to obtain the graph $G(\mathcal{L})$ from the generic partial order $I(\mathcal{L})$.

It should be noted that similar methods were used by Gusfield and Irving [7] to compute efficiently an *egalitarian* solution for the Stable Marriage problem. See also [9], [19] for an application of these methods.

**3.2.1. Constructing the directed graph.** In this section we define a directed graph $G(\mathcal{L})$ such that a one-to-one correspondence can be defined between its closed subsets and the elements of $\mathcal{L}$. (See Fig. 4 for an example.)

Let the set $V_i$ be defined as the set of integers that are contained between the largest and smallest integer feasible values of variable $x_i$. The vertex set of $G(\mathcal{L})$ is $V_1 \cup \cdots \cup V_n$, i.e., a vertex is created for each $v \in V_i$, where $1 \leq i \leq n$. In addition, there is a special vertex denoted by $s$. The edge set of $G(\mathcal{L})$ is defined as follows.

- For each variable $x_i$, a directed chain is defined on the set $V_i$ in sorted order. That is, for each pair of vertices representing two consecutive values, $v$ and $v+1$, there is an arc $(v, v+1)$. Such a chain is called an $x_i$-chain.
- For each inequality $ax_j - bx_i \leq c$, the following "ladder" is defined between the $x_i$-chain and the $x_j$-chain. For all $v \in V_j$, there is an arc from the value corresponding to $\lceil \frac{av-c}{b} \rceil$ in the $x_i$-chain, to the vertex corresponding to $v$ in the $x_j$-chain. Intuitively, the arcs can be thought of as constraints, i.e., if the value of variable $x_j$ is $v$, then the value of variable $x_i$ must be at least $\lceil \frac{av-c}{b} \rceil$ in any feasible solution
- For each $x_i$-chain, there is a bidirectional edge connecting the vertex corresponding to the smallest value in $V_i$ to the vertex $s$. The purpose of these edges is to ensure that any closed set contains at least one vertex in each $x_i$-chain.

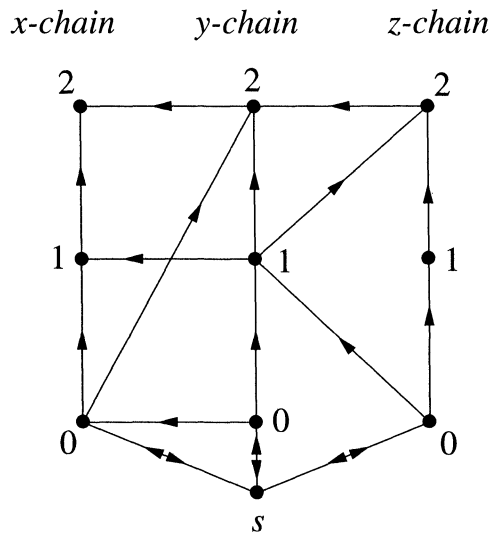


FIG. 4. *The directed graph $G(\mathcal{L})$ corresponding to the set of inequalities of Fig. 3. For example, the arc connecting the "0" value in the x-chain to the "2" value in the y-chain is implied by the inequality $y - x \leq 2$.*

The next theorem states that this construction is valid.

THEOREM 3.6. *There is a one-to-one correspondence between the closed subsets of $G(\mathcal{L})$ that contain vertex s and the elements of $\mathcal{L}$.*

*Proof.* We first prove that every feasible solution vector $L$ defines a closed subset in $G(\mathcal{L})$. Let $L = (u_1, \ldots, u_n)$. The corresponding subset $S_L$ in $G(\mathcal{L})$ is defined by taking for all $i$, all the vertices corresponding to integers that are smaller or equal than $u_i$ in $V_i$, and $s \in S_L$. Assume now that $S_L$ is not a closed subset. Then there exist two vertices, $v \in V_i$ and $w \in V_j$, such that

- $w \in S_L$ and $v \notin S_L$, and,
- there is an arc from $v$ to $w$ in $G(\mathcal{L})$.

This arc can only be generated by the inequality $ax_j - bx_i \leq c$. (We now abuse terminology by treating $v$ and $w$ as both vertices and integers.) Thus, by the construction of $G(\mathcal{L})$, $v = \lceil \frac{aw-c}{b} \rceil$; since $u_i \geq \lceil \frac{au_j-c}{b} \rceil$ and $w \leq u_j$, we get that $v \leq u_i$ and hence $v \in S_L$, contradicting our assumption.

We now prove the other direction. Let $S_L$ be a closed subset in $G(\mathcal{L})$. The solution vector $L = (u_1, \ldots, u_n)$ corresponding to $S_L$ is defined by taking $u_i$ to be the largest integer belonging to $S_L$ in $V_i$. Since $s \in S_L$, then every $x_i$-chain in the graph $G(\mathcal{L})$ has at least one representative in the closed subset $S_L$. Suppose $L$ is infeasible, i.e., there exists an inequality $ax_j - bx_i \leq c$ such that $au_j - bu_i > c$. By the construction of $G(\mathcal{L})$, there is an arc from the vertex in $V_i$ corresponding to $\lceil \frac{au_j-c}{b} \rceil$ to the vertex in $V_j$ corresponding to $u_j$. Since $S_L$ is a closed subset, the vertex corresponding to $\lceil \frac{au_j-c}{b} \rceil$ must belong to $S_L$, and hence, $u_i \geq \lceil \frac{au_j-c}{b} \rceil$, which contradicts the invalidity of the inequality. □

REMARK 3.1. *Notice that $G(\mathcal{L})$ has a succinct description and can be encoded in polynomial space. This implies that the complete set of solutions of a monotone system can be completely encoded in polynomial space.*

**3.2.2. Minimizing with respect to a modular function.** In this section we discuss how to compute a lattice element minimizing an objective function $\sum_{i=1}^{n} w_i x_i$. As mentioned earlier, this objective function also defines a modular function on the lattice. We first show how to assign costs to the vertices of $G(\mathcal{L})$ such that a closed subset of minimum cost corresponds to a lattice element of minimum cost. Then we briefly review Picard's algorithm [16] for finding a closed subset of minimum cost in a directed graph.

The cost of every vertex in $G(\mathcal{L})$ is determined as follows. Let the smallest value in $V_i$ be $b_i$. The cost of the vertex corresponding to $b_i$ is $w_i \cdot b_i$, and the cost of the other vertices in $V_i$ is $w_i$. It is not hard to see that finding the optimal solution with respect to an objective function is equivalent to finding the closed subset of minimum cost in $G(\mathcal{L})$, where the cost of a closed set is defined to be the sum of the costs of its members. The problem of computing the minimum cost closed set can be reduced to computing the minimum cut in the following graph (of pseudo-polynomial size), denoted by $G$, which is derived from $G(\mathcal{L})$. (Computing the minimum cut in $G$ can be done by finding the maximum flow from the source to the sink.)

- Connect all positive cost vertices to a source and all negative cost vertices to a sink.
- The capacity assigned to edges adjacent to the source or sink is equal to the absolute value of the cost of the vertices to which they are adjacent.
- All other edges in $G(\mathcal{L})$ have infinite capacity in $G$.

By our construction, the minimum cut must either contain edges adjacent to the source or to the sink. (Other edges have infinite capacity.) The *sink-set* of a cut is defined to be the set of vertices that can be reached from the source only via the cut. Picard [16] proved that the sink-set defined by the minimum cut in $G$ corresponds to a closed subset of minimum cost in $G(\mathcal{L})$. To see that, let $N$ be the sum of the capacities of the edges adjacent to the source in $G$. It is not hard to see that the cost of the vertices in the sink-set of any finite cut is equal to $-N$ plus the capacity of the cut. Hence, a minimum cut defines a closed subset of minimum cost.

Since a minimum cut can be identified in a graph $G = (V, E)$ in $O(|E||V| \log |V|)$, e.g., [8], and in our graph $|V| = O(\sum_{i=1}^{n} |V_i|)$ and $|E| \leq O(m \sum_{i=1}^{n} |V_i|)$, we have the following theorem.

THEOREM 3.7. *The integer optimal solution of a monotone system of inequalities with respect to an arbitrary linear objective function can be computed in pseudo-polynomial time, in* $O(m(\sum_{i=1}^{n} |V_i|)^2 \log(\sum_{i=1}^{n} |V_i|))$ *time.*

**3.2.3. The generic construction.** In order to motivate the construction of $G(\mathcal{L})$, we now present without proof how it can be obtained from the generic partial order $I(\mathcal{L})$.

Let $\mathcal{L}[x_i = a]$ denote the set of all feasible solution vectors for which $x_i = a$. Obviously, $\mathcal{L}[x_i = a]$ induces a sublattice of $\mathcal{L}$. We call a lattice element *irreducible* if for some variable $x_i$ and integer $a$, it is the bottom element of the sublattice $\mathcal{L}[x_i = a]$. The partial order $(I(\mathcal{L}), \preceq)$ is defined as follows: the vertex set is the set of irreducible elements of the lattice $\mathcal{L}$; for elements $a, b \in I(\mathcal{L})$, there is an edge from $a$ to $b$, if $a \preceq b$ in $\mathcal{L}$. The following theorem is proved in [6, p. 72, Thm. 9] and [7, Thm. 2.2.1].

THEOREM 3.8. *There is a one-to-one correspondence between the nonempty closed subsets of* $I(\mathcal{L})$ *and the elements of* $\mathcal{L}$. *Moreover, if closed subsets* $S$ *and* $S'$ *of* $I(\mathcal{L})$ *correspond to vectors* $L$ *and* $L'$, *respectively, then* $L'$ *dominates* $L$ *if and only if* $S \subseteq S'$.

However, the partial order $I(\mathcal{L})$ has a "complicated" structure which we now show how to simplify and make more regular. (This generalizes the construction in [10].)

The elements $L_1$ and $L_2$ are called *consecutive* elements in the lattice $\mathcal{L}$ if $L_2$ covers $L_1$, i.e., there is no element $M$ such that $L_1 < M < L_2$. Suppose elements $L_1$ and $L_2$ are consecutive and $L_1 < L_2$. The *minimal difference* between $L_1$ and $L_2$ is defined to be the "set of changes" between $L_1$ and $L_2$. More formally, by a single *change* we mean the difference between the value of a variable in $L_1$ and $L_2$. We denote by $\mathcal{D}$ the set of all minimal differences in $\mathcal{P}$.

A *maximal chain* in a lattice is a chain of consecutive elements that starts at $B$ and ends at $T$. An interesting property of distributive lattices is that each maximal chain contains *all* the minimal differences. The minimal differences appear on each maximal chain in some order and each minimal difference appears exactly once.

We can now define the partial order $(T(\mathcal{L}), \preceq)$. Let $D_1, D_2 \in \mathcal{D}$; then $D_1 < D_2$ if and only if $D_1$ precedes $D_2$ on every maximal chain in $\mathcal{P}$. We are now ready for the next theorem, whose proof follows from [7, Thm. 2.4.4] and which relates the partial orders $I(\mathcal{L})$ and $T(\mathcal{L})$.

THEOREM 3.9. *There is a one-to-one correspondence between the closed subsets of* $I(\mathcal{L})$ *and* $T(\mathcal{L})$.

In fact, the partial order $T(\mathcal{L})$ is very similar to $G(\mathcal{L})$. Let $\hat{V}_i$ denote the set of integer feasible values of variable $x_i$. Notice that the elements of $\hat{V}_i$ do not necessarily form a consecutive interval, in contrast to the fractional case, where all values $\lambda$ such that $x_i^{\min} \leq \lambda \leq x_i^{\max}$ are feasible. Notice that $T(\mathcal{L})$ is the structure we obtain if we follow the definition of $G(\mathcal{L})$ except that the set $V_i$ is replaced by $\hat{V}_i$. For example, if $\hat{V}_i = \{1, 5, 6\}$, then there is an arc from the vertex corresponding to "1" to the vertex corresponding to "5", and an arc from the vertex corresponding to "5" to the vertex corresponding to "6."

The difficulty in constructing the partial order $T(\mathcal{L})$ is that we need to generate the elements of the sets $\hat{V}_i$ one by one, since they are not necessarily sets of consecutive integers. This can be done in pseudo-polynomial time; however, $T(\mathcal{L})$ does not have a succinct description which motivates the construction of $G(\mathcal{L})$.

**4. Identifying fat polytopes.** This section presents an application of the Fourier–Motzkin algorithm for identifying *fat* polytopes.

Even though it is NP-hard to decide whether a set of inequalities has an integer feasible solution, one can use a fast preprocessing stage to compute an integer feasible solution in certain cases. This preprocessing stage runs in strongly polynomial time for the case of linear programs with two variables per inequality. It checks whether the polytope is *fat*, i.e., whether

it contains a sphere circumscribing a unit hypercube. Since a unit hypercube must contain at least one integer lattice point, an integer feasible point is found by rounding the coordinates of the center of the sphere to the nearest integer. This procedure is a heuristic for finding a feasible integer point, since there may exist a feasible integer point in the polytope, yet the polytope does not contain a large enough sphere. Lenstra [12] uses a similar procedure that works in polynomial time and may identify a feasible integer point; however, in his procedure the running time depends on the ellipsoid method and is therefore not strongly polynomial.

The idea of the procedure is to shift all constraints by a distance of $r$. Any feasible point in the resulting set of inequalities is at a distance of $r$ from all the faces of the polytope, and hence a sphere of radius $r$ around any such feasible point is contained in the polytope. In order to obtain a sphere large enough to contain a unit hypercube, we need to set $r = \sqrt{n}/2$.

Shifting a constraint by a distance $r$ is done as follows. Given an inequality $\sum_{i=1}^{n} a_i x_i \leq c$, the shifted inequality is $\sum_{i=1}^{n} a_i x_i \leq \bar{c}$, where

$$\bar{c} = c - \frac{r}{\sqrt{\sum_{i=1}^{n} a_i^2}} \sum_{i=1}^{n} a_i .$$

In the case of two variables per inequality, the sum includes at most two terms. The new set of inequalities, each with a constant $c$ substituted by $\bar{c}$, is also a set of inequalities with two variables per inequality, and hence is solvable in the running time reported in §2. Consequently, we can test whether a polytope is *fat* and find an integer feasible point in $O(mn^2 \log m)$ time.

Although the running time is polynomial, the procedure involves the manipulation of square roots, which may be difficult in practice. Since this algorithm only finds a feasible integer point in a special case, it is satisfactory for this purpose to truncate $\bar{c}$ to a small number of accuracy bits, where the small number depends on the machine word length or on other implementation considerations.

An interesting related problem is to find the *largest* sphere contained in a polytope. For this we need to maximize the value of $r$ such that the system still has a feasible solution. Although such a problem has three variables per inequality, it is still solvable in strongly polynomial time. This follows from the results of [3] and [13], where it is shown that a problem can be solved in strongly polynomial time if, by deleting a constant number of columns (in this case the constant is equal to one), it can be converted to a problem which is solvable in strongly polynomial time. Since treating $r$ as a variable adds only one more variable to the problem, the problem of finding a largest sphere in the polytope defined by a set of inequalities with two variables per inequality is solvable in strongly polynomial time.

**Acknowledgments.** We thank the two anonymous referees for clarifying the presentation of the paper. We thank Arik Tamir for pointing out that the running time of the integer feasibility algorithm is pseudo-polynomial only in the case where the variables are bounded. We would also like to thank Edith Cohen for helpful remarks. Many thanks to Yossi Friedman for his help in making the figures.

## REFERENCES

[1] B. Aspvall and Y. Shiloach, *Polynomial time algorithm for solving systems of linear inequalities with two variables per inequality*, SIAM J. Comput., 9 (1980), pp. 827–845.

[2] E. Cohen, *Combinatorial Algorithms For Optimization Problems*, Ph.D. thesis, Stanford Technical Report, STAN-CS-91-1366, June 1991.

[3] E. Cohen and N. Megiddo, *Strongly polynomial time and NC algorithms for detecting cycles in periodic graphs*, J. Assoc. Comput. Mach., 40 (1993), pp. 791–830.

[4] E. COHEN AND N. MEGIDDO, *Improved algorithms for linear inequalities with two variables per inequality*, Proceedings of the Twenty Third Symposium on Theory of Computing, New Orleans, 1991, pp. 145–155. SIAM J. Comput., this issue, pp. 1313–1347.

[5] H. EDELSBRUNNER, G. ROTE, AND E. WELZL, *Testing the necklace condition for shortest tours and optimal factors in the plane*, Theoret. Comput. Sci., 66 (1989), pp. 157–180.

[6] G. GRÄTZER, *Lattice Theory: First Concepts and Distributive Lattices*, W. H. Freeman and Company, San Francisco, 1971.

[7] D. GUSFIELD AND R. W. IRVING, *The Stable Marriage Problem*, MIT Press, Cambridge, MA, 1989.

[8] A. V. GOLDBERG AND R. E. TARJAN, *A new approach to the maximum flow problem*, J. Assoc. Comput. Mach., 35 (1988), pp. 921–940.

[9] M. IRI, *Structural theory for the combinatorial systems characterized by submodular functions*, in Progress in Combinatorial Optimization, Academic Press, New York, 1984, pp. 197–219.

[10] S. KHULLER, J. NAOR, AND P. N. KLEIN, *The lattice structure of flow in planar graphs*, SIAM J. Disc. Math., 6 (1993), pp. 477–490.

[11] J. C. LAGARIAS, *The computational complexity of simultaneous diophantine approximation problems*, SIAM J. Comput., 14 (1985), pp. 196–209.

[12] H. W. LENSTRA JR., *Integer programming with a fixed number of variables*, Math. of Oper. Res., 8 (1983), pp. 538–548.

[13] C. HAIBT-NORTON, S. PLOTKIN, AND E. TARDOS, *Using separation algorithms in fixed dimension*, Proceedings of the First Symposium On Discrete Algorithms, San Francisco, 1990, pp. 377–387.

[14] N. MEGIDDO, *Towards a genuinely polynomial algorithm for linear programming*, SIAM J. Comput., 12 (1983), pp. 347–353.

[15] C. G. NELSON, *An $n^{\log n}$ algorithm for the two-variable-per-constraint linear programming satisfiability problem*, Technical Report AIM-319, Stanford University, 1978.

[16] J. C. PICARD, *Maximal closure of a graph and applications to combinatorial problems*, Management Sci., 22 (1976), pp. 1268–1272.

[17] A. SCHRIJVER, *Theory of Linear and Integer Programming*, John Wiley, New York, 1986.

[18] R. SHOSTAK, *Deciding linear inequalities by computing loop residues*, J. Assoc. Comput. Mach., 28 (1981), pp. 769–779.

[19] D. TOPKIS, *Minimizing a submodular function on a lattice*, Oper. Res., 26 (1978), pp. 305–321.

[20] A. F. VEINOTT, *Representation of general and polyhedral subsemilattices and sublattices of product spaces*, Linear Algebra Appl., 114/115 (1989), pp. 681–704.