SIAM J. SCI. STAT. COMPUT.
Vol. 13, No. 5, pp. 1194–1217, September 1992

NEW BRANCH-AND-BOUND RULES FOR LINEAR BILEVEL PROGRAMMING*

PIERRE HANSEN†, BRIGITTE JAUMARD‡, AND GILLES SAVARD§

Abstract. A new branch-and-bound algorithm for linear bilevel programming is proposed. Necessary optimality conditions expressed in terms of tightness of the follower's constraints are used to fathom or simplify subproblems, branch and obtain penalties similar to those used in mixed-integer programming. Computational results are reported and compare favorably to those of previous methods. Problems with up to 150 constraints, 250 variables controlled by the leader, and 150 variables controlled by the follower have been solved.

Key words. bilevel programming, Stackelberg game, variable elimination, branch and bound

AMS(MOS) subject classifications. 90C27, 90D05

1. Introduction. In many situations, multiple decision makers with divergent objectives intervene in the decisions to be made (Wendell [38]). The simplest such case, in which there are only two decision makers, has long been studied in game theory. If there is some asymmetry between the decision makers in that one of them, called the *leader*, makes his decisions first, anticipating the reaction of the other one, called the *follower*, and cooperation is ruled out a priori, one has a Stackelberg game. Adding joint constraints on the strategies of the leader and the follower makes the model more realistic. This leads to *bilevel programming*, a topic which has attracted much attention recently, mostly in the linear case (see Fortuny-Amat and McCarl [19]; Candler and Townsley [16]; Papavassilopoulos [34]; Bard [5]; Bialas and Karwan [10], [11]; Ünlü [37]; Dempe [17]; Bard and Moore [9]; Judice and Faustino [27], [28]; Ben-Ayed and Blair [12]; Haurie, Savard, and White [24]; Anandalingam and White [4]). The nonlinear case and extensions to three or more levels are treated in a few papers (e.g., Aiyoshi and Shimizu [1], [2]; Bard [6], [7]; Bard and Falk [8]; Bard and Moore [9]; Al-Khayyal, Horst, and Pardalos [3]). Related bilevel and multilevel decision models and algorithms have also been extensively studied in control theory.

Applications of bilevel programming have been made in many domains; these include economic development policy (Candler and Norton [15]), agricultural economics (Candler, Fortuny-Amat, and McCarl [14]), road network design (Marcotte [30], Leblanc and Boyce [29]), and cogeneration of electrical energy (Savard [35], Haurie, Loulou, and Savard [23]). Mathematical programs with optimization problems in the constraints (e.g., Bracken and McGill [13]), arising in weapons allocation problems, can also be viewed as (nonlinear) bilevel programs.

The paper is organized as follows. In §2, the linear bilevel programming problem is stated mathematically. It is shown to be strongly NP-hard in §3. Our algorithm is

* Received by the editors May 1, 1989; accepted for publication (in revised form) July 2, 1991.

† Rutgers Center for Operations Research, Rutgers University, Hill Center for the Mathematical Sciences, New Brunswick, New Jersey 08903. The work of this author was supported in part by Air Force Office of Scientific Research grants 0271 and 0066 to Rutgers University, and done in part during two visits to GERAD, Montréal.

‡ Groupe d'Études et de Recherche en Analyse des Décisions and École Polytechnique de Montréal, Case Postale 6079, Succursale "A," Montréal, Canada, H3C 3A7. The work of this author was supported by a Natural Sciences and Engineering Research Council of Canada grant GP0036426 and a Fonds pour la Formation de Chercheurs et l'Aide à la Recherche of Québec grant 90NC0305.

§ Groupe d'Études et de Recherche en Analyse des Décisions and Collège Militaire Royal de Saint-Jean, Saint-Jean-sur-Richelieu, Canada, J0J 1R0. The work of this author was supported by Programme de Recherche Universitaire du Ministère Canadien de la Défense grant FUHBO.

stated in §4 and illustrated in an example in §5. Computational experience is reported in §6.

2. Formulation. The linear bilevel program (LBP) may be written in the following general form:

$$\begin{aligned}
&(1) \qquad \max_{x}\text{imize} \quad z_1(x, y(x)) = c^1x + d^1y(x)\\
&\qquad\qquad \text{subject to:}\\
&(2) \qquad A^1x + B^1y(x) \le b^1,\\
&(3) \qquad x \ge 0;\\
&(4) \qquad \max_{y}\text{imize} \quad z_2(x, y) = c^2x + d^2y,\\
&\qquad\qquad \text{subject to:}\\
&(5) \qquad A^2x + B^2y \le b^2,\\
&(6) \qquad y \ge 0,
\end{aligned}$$

where

$$\begin{array}{lll}
c^{1T}, c^{2T}, x \in \mathbb{R}^{n_1}; & d^{1T}, d^{2T}, y \in \mathbb{R}^{n_2}; & A^1 \in \mathbb{R}^{m_1} \times \mathbb{R}^{n_1};\\
A^2 \in \mathbb{R}^{m_2} \times \mathbb{R}^{n_1}; & B^1 \in \mathbb{R}^{m_1} \times \mathbb{R}^{n_2}; & B^2 \in \mathbb{R}^{m_2} \times \mathbb{R}^{n_2};\\
b^1 \in \mathbb{R}^{m_1}; & b^2 \in \mathbb{R}^{m_2}. &
\end{array}$$

For simplicity, we assume the polyhedra defined by (3), (5), and (6) and by (2), (3), (5), and (6) to be nonempty and bounded. The constraints on $y(x)$ implicitly defined by (4)–(6) can be represented as

$$(7) \qquad y(x) \in \arg\max\{z_2(x, y)\,;\, y \in \Omega^2(x)\},$$

where

$$(8) \qquad \Omega^2(x) = \{y \,:\, B^2y \le b^2 - A^2x, y \ge 0\}.$$

A solution $(x, y(x))$ such that $y(x)$ satisfies (7) and (8) is called *rational*. The set of rational solutions of LBP is, in general, not convex and, in the presence of first-level constraints (2), may be empty (in which case LBP has no solution). If the set of rational solutions is nonempty, at least one optimal solution of LBP is obtained at an extreme point of the polytope defined by (2), (3), (5), and (6). This important property was proven by Candler and Townsley [16], Bialas and Karwan [10], and Bard [5]. (Their proofs are for the case in which there are no first-level constraints (2), but readily extend to the case where such constraints are present.)

The leader controls the variables of x, and the follower, those of y, which are chosen after those of x; i.e., by solving the follower's subproblem (4)–(6). The model (1)–(6), due to Aiyoshi and Shimizu [1], contains first-level constraints that are binding only for the leader but depend also on the decisions of the follower. Such constraints arise, for instance, when the leader, a central agency, decides upon public investments and has to bear the cost of their maintenance, which depends on the level of use, decided upon by the follower, i.e., the users. Another instance is the problem of cogeneration of electricity by the leader, a central agency, and by the follower, i.e., independent firms (Savard [35], Haurie, Loulou, and Savard [23]). Electricity produced by the latter is bought at marginal cost. If the constraint on demand satisfaction is introduced at the second level, the optimal decision of the central agency is to produce no electricity at all, which is clearly unrealistic. Introducing the demand constraint at the first level requires the leader to produce a sufficient amount of electricity for the remaining demand to be satisfied through voluntary decisions of the follower.

Some recent algorithms for LBP, e.g., those of Bard and Moore [9], which do not allow for y variables in the first-level constraints, can easily be modified to handle them (see §6).

Algorithms for linear bilevel programming may be grouped as follows:

(i) Branch-and-bound algorithms, in which branching is done on complementarity conditions (Fortuny-Amat and McCarl [19], Bard and Falk [8], Bard and Moore [9]).

(ii) Extreme point-ranking methods (Papavassilopoulos [34], Bialas and Karwan [11]).

(iii) Algorithms based on finding efficient solutions for the leader's and follower's objective functions (Bard [5], Ünlü [37]). These algorithms should be viewed as heuristics, as the set of efficient solutions may contain no optimal solution (see, e.g., Haurie, Savard, and White [24]).

(iv) Algorithms using complementary pivoting (Candler and Townsley [16]; Bialas and Karwan [11]; Judice and Faustino [27], [28]). The algorithm of Bialas and Karwan [11] does not always provide an optimal solution, as shown in Judice and Faustino [27]; the latter authors' parametric method provides an ε-optimal solution for any given ε.

(v) Penalty function methods (Dempe [17], Anandalingam and White [4]).

(vi) Reverse convex programming (Al-Khayyal, Horst, and Pardalos [3]).

The best computational results to date appear to be those of Bard and Moore [9] and of Judice and Faustino [27], [28].

In this paper we propose a new algorithm of class (i). It exploits necessary conditions for subsets of the follower's constraints to contain at least one constraint that is tight at the optimum. These conditions are new in linear bilevel programming, although similar ones have been used in global optimization in design. They allow fathoming of subproblems for which they cannot be satisfied and lead to better bounds, as well as penalties, similar to those used in mixed-integer programming. We also investigate many branching rules, some of them based on logical relations expressing the conditions on the tightness of constraints that have been detected.

For fixed $\overline{x}$ the follower's subproblem (4)–(6) has a single optimal solution $\overline{y}$ in the absence of dual degeneracy. Otherwise there is an infinity of optimal $\overline{y}$'s, not all of which need to be feasible or of equal value in the leader's problem. Then some further assumptions have to be made on the willingness of the follower to cooperate with the leader in order for the problem to be well defined. From the point of view of the leader, the best case is when the follower accepts the leader's preferences regarding the y_j's for which he (the follower) is indifferent (the tie cooperative case) and the worst case is when the follower adopts the opposite of the leader's preferences regarding these y_j's (the tie noncooperative case). In this paper, we present an algorithm for the tie cooperative case, and indicate briefly how it can be modified to solve the tie noncooperative case.

As cooperation between the leader and the follower is ruled out, except in the case of ties for the follower's objective function, the optimal solution of LBP need not be an efficient one. In other words, there might be another feasible solution with higher value for both objective functions.

The LBP includes as particular cases the *linear max-min* (LMM) problem and the bilinear programming problem and is equivalent to some cases of concave programming and of nonconvex quadratic programming (see Pardalos and Rosen [33] for definitions and references).

3. Complexity.

Jeroslow [26] has shown that LBP is NP-hard, and a shorter proof has recently been given by Ben-Ayed and Blair [12]. We next show that this result can be strengthened; we consider the LMM problem, a more restricted model than LBP. LMM is obtained from LBP by omitting constraints (2) and setting $c^2 = -c^1$, $d^2 = -d^1$, as pointed out by Bard and Falk [8]. For definitions, see Garey and Johnson [20].

THEOREM 3.1. LMM *is strongly* NP*-hard.*

Proof. We use reduction to KERNEL, which has been proved to be NP-hard by Chvátal (see Garey and Johnson [20, p. 204]). Recall that a kernel K of a graph $G = (V, E)$ is a vertex set that is stable (no two vertices of K are adjacent) and absorbing (any vertex not in K is adjacent to a vertex of K). Then consider the LMM

$$\max_x \min_y z = -\sum_{j=1}^{n} y_j, \tag{9}$$

$$\text{s.t.} \quad x_j + x_k \le 1 \qquad \forall\, j,k | \{v_j,\, v_k\} \in E, \tag{10}$$

$$y_j \le 1 - x_j \quad \forall\, j, \tag{11}$$

$$y_j \le 1 - x_k \quad \forall\, j,k | \{v_j,\, v_k\} \in E, \tag{12}$$

$$x_j, y_j \ge 0 \qquad \forall\, j, \tag{13}$$

where variables x_j and y_j are both associated with the vertex v_j of G, for $j = 1, 2, \cdots, n$. We claim that G has a kernel if and only if the optimal solution to (9)–(13) has a value $z^* = 0$. Assume first that G has a kernel K. Then set $x_j = 1$ for all $j \,|\, v_j \in K$, $x_j = 0$ otherwise. This solution satisfies (10) as K is stable. It imposes $y_j = 0$ for all j through (11) and (12) as K is absorbing; so $z^* = 0$.

Assume next that G has no kernel. Then any optimal solution in which all x_j^*'s are integers defines a stable set $S = \{v_j \,|\, x_j^* = 1\}$. As this stable set cannot be absorbing, at least one y_j^* must be equal to 1 and $z^* \le -1$. Moreover, any nonintegral optimal solution must contain at least one x_j^* taking a positive fractional value, and because of (10) no x_k^* such that $(v_j,\, v_k) \in E$ can be equal to 1. So y_j^* takes a positive value and $z^* < 0$. As KERNEL is not a number problem, this completes the proof. □

COROLLARY 3.2. LBP *is strongly* NP*-hard.*

Recall that a *fully polynomial approximation scheme* provides an ε-optimal solution to a given problem in time polynomial in the problem size and in ε. It follows from the above corollary that no fully polynomial approximation scheme can be found for LBP unless $P = NP$.

4. Algorithm.

4.1. Necessary conditions for optimality and penalties.

We next derive necessary optimality conditions, expressed in terms of tightness of constraints in the follower's subproblem, i.e., constraints (5) and (6). To this effect we associate with each such constraint a boolean variable α_i equal to 1 if the constraint is tight, and equal to 0 otherwise. These conditions will play a basic role in the branch-and-bound algorithm described in §4.2

THEOREM 4.1. *In any rational solution to* LBP *the tightness of the constraints in the follower's subproblem is such that*

$$\sum_{i|B^2_{ij}>0} \alpha_i \geq 1 \quad \text{if } d^2_j > 0, \tag{14}$$

$$\sum_{i|B^2_{ij}<0} \alpha_i + \alpha_{m_2+j} \geq 1 \quad \text{if } d^2_j < 0 \tag{15}$$

for $j = 1, 2, \cdots, n_2$.

Proof. Assume that by contradiction there is a rational solution $(\hat{x}, \hat{y})$ to LBP such that (14) does not hold for some $j \in \{1, 2, \cdots, n_2\}$ such that $d_j > 0$ or (15) does not hold for some $j \in \{1, 2, \cdots, n_2\}$ such that $d_j < 0$. In the former case, increasing the value of $\hat{y}_j$ by

$$\Delta\hat{y}_j = \min_{i|B^2_{ij}>0} \left\{ \frac{1}{B^2_{ij}} \cdot \left(b^2_i - \sum_{k=1}^{n_1} A^2_{ik}\hat{x}_k - \sum_{\substack{k=1\\k\neq j}}^{n_2} B^2_{ik}\hat{y}_k \right) \right\} \tag{16}$$

yields a solution satisfying constraints (5) for $x = \hat{x}$ and (6) with a value larger by $d_j\Delta\hat{y}_j$ than that of $(\hat{x}, \hat{y})$. This contradicts $(\hat{x}, \hat{y})$ being rational. In the latter case, decreasing the value of $\hat{y}_j$ by

$$\Delta\hat{y}_j = \min \left[\hat{y}_j, \min_{i|B^2_{ij}<0} \left\{ \frac{1}{B^2_{ij}} \left(\sum_{k=1}^{n_1} A^2_{ik}\hat{x}_k + \sum_{\substack{k=1\\k\neq j}}^{n_2} B^2_{ik}\hat{y}_k - b^2_i \right) \right\} \right] \tag{17}$$

yields a solution satisfying constraints (5) for $x = \hat{x}$ and (6) with a value larger by $-d_j\Delta\hat{y}_j$ than that of $(\hat{x}, \hat{y})$. This again contradicts $(\hat{x}, \hat{y})$ being rational. □

COROLLARY 4.2. *In any optimal solution to* LBP *the tightness of the constraints in the follower's subproblem is such that conditions* (14) *and* (15) *are satisfied for all* $j \in \{1, 2, \cdots, n_2\}$ *such that* $d_j > 0$ *and* $d_j < 0$, *respectively.*

Proof. This follows immediately from the fact that optimal solutions to LBP are rational. □

A weaker condition involving tightness of constraints was obtained by Falk [18] for the LMM problem. Falk indeed noted that at least one constraint in the follower's subproblem must be tight at the optimum, i.e., that

$$\sum_{i=1}^{m_2} \alpha_i + \sum_{j=1}^{n_2} \alpha_{m_2+j} \geq 1. \tag{18}$$

Falk [18] showed that using this relation at the first node while solving by a branch-and-bound algorithm does reduce computational effort, but he did not apply it for subproblems obtained by branching. We next discuss branching for the LBP. This leads us to show how further relations of type (14) or (15) can be obtained for all subproblems.

In all rules studied, branching is done by fixing some binary variable(s) α_i at 0 or at 1. Either a single variable α_i will be chosen for that purpose (dichotomous branching), or a logical relation (14) or (15), e.g., $\alpha_{i_1} + \alpha_{i_2} + \alpha_{i_3} + \cdots \geq 1$ will be chosen and branching will be done according to the rule $\alpha_{i_1} = 1$ or ($\alpha_{i_1} = 0$ and $\alpha_{i_2} = 1$) or ($\alpha_{i_1} = \alpha_{i_2} = 0$ and $\alpha_{i_3} = 1$) and so on (multiple branching).

If $\alpha_i = 1$, the ith constraint (in (5) or (6)) in the follower's subproblem becomes an equality. It can then be used to eliminate one of the follower's variables y_j (which is y_{i-m_2} in case of a constraint of type (6)). New logical relations of type (14) and (15) can then be derived.

Of course, variable elimination does not reduce the number of structural constraints in that the nonnegativity constraint $y_j \geq 0$ for the eliminated variable is replaced by an inequality involving several variables. If there are several possible choices for y_j, the variable with the smallest fill-in, i.e., which least augments the number of nonzero coefficients, is chosen.

If a subproblem is obtained in which no more y variables remain, its optimal solution is found by solving the problem obtained by deleting the objective function (4) of the follower (called leader relaxation below and denoted LR), as the leader controls all remaining variables. However, it will be necessary to check whether this solution is rational or not. This will be the case if the value of the follower's objective function, when the x variables are fixed at their values in the optimal solution of LR, is the same for the y_j values obtained from the tight constraints used to eliminate them, and for the optimal y_j values for the follower's subproblem.

If $\alpha_i = 0$, the ith constraint (15) in the follower's subproblem becomes a strict inequality, and from the complementary slackness theorem of linear programming, the ith variable in the dual of the follower's subproblem must be equal to 0. As it is not easy to handle constraints stating that a variable is strictly positive in linear programming, the dual of the follower's subproblem will be solved instead of the primal in one test of the algorithm described below. When many variables α_i are fixed at 0, this dual problem may be infeasible, in which case, from the duality theorem, the primal follower's subproblem together with the strict positivity constraints is also infeasible, as by assumption it is bounded.

In practice, variable elimination can be performed by pivoting in a simplex tableau and fixing the eliminated variable, which leaves the basis at 0. This is easily implemented in a linear programming package such as Marsten's XMP [31].

In the branch-and-bound algorithm described in the next subsection, depth-first search is used and the subset of logical relations (denoted by R) is updated after either branching or backtracking.

When a branch corresponding to $\alpha_i = 0$ is explored, all logical relations involving α_i are simplified by deleting α_i. When a branch corresponding to $\alpha_i = 1$ is explored, all logical relations involving α_i are deleted as they are trivially satisfied. Then new relations (14) or (15) are obtained after a variable y_j has been eliminated. Finally, redundant relations are eliminated from R. (Recall that a logical relation $r_k \equiv \sum_{i \in I_k} \alpha_i \geq 1$, where I_k denotes the set of indices of the logical variables in r_k, of R is redundant if R contains another logical relation $r_\ell \equiv \sum_{i \in I_\ell} \alpha_i \geq 1$ such that $I_\ell \subseteq I_k$, i.e., satisfaction of $r_\ell \geq 1$ implies that $r_k \geq 1$ is satisfied as all variables α_i appearing in r_ℓ appear also in r_k.) When backtracking occurs, we revert to the set R corresponding to the last explored node for which one branch is unexplored and then explore this branch, updating R accordingly.

When branching is done on the values of the α_i, at most $2^{m_2+n_2+1}-1$ subproblems will be generated. If multiple branching is used the number of subproblems will be less, as subproblems with $\alpha_i = 0$ for all $i \in I_k$ for relations $r_k \in R$ used for branching are excluded.

As in many other algorithms for LBP, linear programming will be used to obtain bounds on the optimal value. To this effect, the objective functions (4) of the follower will be deleted (in the original problem or in the current subproblem in which some variables y_j have been eliminated). Solving the so-obtained linear program LR gives such an upper bound (denoted z_L^*). Then the effect of fixing a variable α_i at 1, i.e., satisfying a constraint as an equality, can be anticipated to some extent by computing a penalty as in mixed-integer programming (see, e.g., Taha [36]). Consider the equations corresponding to an optimal tableau of the LR of the current subproblem

$$z = z_L^* - \sum_{j \in N} (z_j^* - c_j)x_j,$$

$$x_i = b_i^* - \sum_{j \in N} A_{ij}^* x_j, \qquad i \in B,$$

where B denotes the index set of basic and N the index set of nonbasic variables. Then if x_i is the slack variable of the ith constraint the down penalty for setting x_i at 0 is

$$p_i = b_i^* \min_{j \in N | A_{ij}^* > 0} \left\{ \frac{z_j^* - c_j}{A_{ij}^*} \right\}.$$

It corresponds to the decrease in the value of z_L during the first dual-simplex iteration after adding the constraint $x_i \leq 0$. Moreover, if $r_k \equiv \sum_{i \in I_k} \alpha_i \geq 1$ is a logical relation of type (14) or (15) that must be satisfied by any rational solution, then at least one of the slack variables x_i ($i \in I_k$) must be set at 0. It follows that

$$z_L^{*'} = z_L^* - \min_{i \in I_k} p_i$$

is an upper bound on the optimal value of the current subproblem. Finally, taking into account all logical relations of type (14) or (15) leads to a stronger upper bound:

$$z_L^{*''} = z_L^{*'} - \max_{k | r_k \in R} \min_{i \in I_k} p_i.$$

4.2. Tests. Before giving the rules of the new algorithm, we recall and illustrate a classification of tests for branch-and-bound methods (Hansen, Jaumard, and Lu [21], [22]). Viewed abstractly, any test of a branch-and-bound algorithm consists in examining whether a sufficient condition for some proposition about the current subproblem to be true is satisfied or not, and making use of this proposition when it is the case to fathom or simplify the subproblem.

A *direct test* is such that the information provided by the proposition when the condition holds is all that is required for the solution of the current subproblem. This is the case when it can be shown: (i) that a known solution x^* is the globally optimal solution of the subproblem (*direct resolution test*); or (ii) that the subproblem has no solution better than the best one known, called *incumbent* (*direct optimality test*); or (iii) that the subproblem has no feasible solution (*direct feasibility test*).

One resolution test for LBP, which applies to subproblems in which all variables y_j have been eliminated, consists in solving LR for x_L^* and then checking whether the solution (x_L^*, y_L^*), where y_L^* is obtained by using the equations of elimination backwards, is rational or not. If it is rational, (x_L^*, y_L^*) is the optimal solution of the current subproblem. A stronger version of this test, in which some y_j may remain in LR, is given below. One direct optimality test for LBP consists of solving LR and checking if its optimal value z_L^* exceeds the value z_{opt} of the best solution found so far. One direct feasibility test for LBP, discussed above, consists in checking whether the dual of the follower's relaxation (FR) is feasible or not.

A *conditional test* is such that the information provided by the proposition when the condition holds is all that is required for the solution of the current subproblem when a variable is fixed at a given value (here a boolean variable α_i fixed at the value 1).

One conditional optimality test for LBP consists in solving LR, computing the penalties p_i for loose constraints (i.e., those with strictly positive slack variables in the basis) and checking if $z_L^* - p_i \leq z_{\text{opt}}$. If this last inequality holds, no better solution than the incumbent one can be found for the current subproblem with the ith constraint being tight (i.e., $\alpha_i = 1$), so α_i may be fixed at 0.

A *relational test* is such that the information provided by the proposition when the condition holds is all that is required for the solution of the current subproblem if some algebraic or (here) logical relation is not satisfied.

Applying Theorem 4.1 yields logical relations (14) and (15) for all y_j remaining in the current subproblem; if all α_i in one of these relations are equal to 0, the current subproblem contains no rational solution and is fathomed.

We next describe a depth-first branch-and-bound algorithm. The current subproblem is characterized by: (i) objective functions and constraints of type (1)–(6) in x and in the remaining y_j variables; (ii) the vector α specifying which of the initial constraints are tight, loose, or of unknown tightness; (iii) the logical relations obtained from monotonicity; and (iv) the list of eliminated variables and the equalities defining their values. This subproblem may be written as follows:

$$\max_x \quad \tilde{z}_1(x, \tilde{y}) = \tilde{c}^1 x + \tilde{d}^1 \tilde{y}, \tag{19}$$

subject to:

$$\tilde{A}^1 x + \tilde{B}^1 \tilde{y} \leq \tilde{b}^1, \tag{20}$$

$$x \geq 0; \tag{21}$$

$$\max_{\tilde{y}} \quad \tilde{z}_2(x, \tilde{y}) = \tilde{c}^2 x + \tilde{d}^2 \tilde{y}, \tag{22}$$

subject to:

$$\tilde{A}^2 x + \tilde{B}^2 \tilde{y} \leq \tilde{b}^2, \tag{23}$$

$$\tilde{y} \geq 0, \tag{24}$$

where $\tilde{y}$ denotes the vector of remaining y_j variables and the tilde indicates that the coefficients in (19)–(24) are those obtained by eliminating the variables of $y \setminus \tilde{y}$. In addition, the current subproblem is defined by $\alpha = (\alpha_1, \cdots, \alpha_{m_2+n_2})$, where the α_i can be fixed at 0 or at 1 or be free, and $R = \{r_k, k \in K\}$, where $r_k \equiv \sum_{i \in I_k} \alpha_i \geq 1$ with $I_k \subseteq \{1, 2, \cdots, m_2 + n_2\}$, as well as the linear equations used to eliminate the variables of $y \setminus \tilde{y}$.

As discussed above, elimination of variables can be done by pivoting and fixation of nonbasic variables at 0. For simplicity of exposition we will not distinguish between eliminated and noneliminated variables when stating the rules of the algorithm given below.

We use two relaxations of the subproblem: the *leader's relaxation* (LR) obtained, as explained above, by omitting (22), and the *follower's relaxation* (FR), which consists of (22)–(24) for x fixed to $\overline{x}$. We will also consider the *follower's subproblem* (FS), which consists of (4)–(6), for $x = \overline{x}$.

ALGORITHM HJS

a. Initialization.
Obtain an initial solution (x_h, y_h) with a heuristic (the choice of which is discussed below). Initialize the incumbent solution $(x_{\text{opt}}, y_{\text{opt}})$ to (x_h, y_h) and the incumbent value z_{opt} to $c^1 x_{\text{opt}} + d^1 x_{\text{opt}}$. If no heuristic solution can be found, initialize $(x_{\text{opt}}, y_{\text{opt}})$ to an arbitrary value and set $z_{\text{opt}} = -\infty$.
Consider all variables α_i $(i = 1, 2, \cdots, m_2 + n_2)$ to be free. Set $R = \emptyset$.

b. First direct optimality test.
Solve LR; let (x_L^*, y_L^*) denote an optimal solution.
If $z_L^* = c^1 x_L^* + d^1 y_L^* \le z_{\text{opt}}$, go to step m (backtracking).

c. First direct feasibility test.
Solve the dual of FR(x). If it has no feasible solution, go to step m.

d. Direct resolution test, first part.
Consider again the optimal solution (x_L^*, y_L^*) of LR.
Check if (x_L^*, y_L^*) is rational for the current subproblem: solve FR(x_L^*), and let y_F^* be an optimal solution. If $d^2 y_L^* = d^2 y_F^*$ then (x_L^*, y_L^*) is rational; otherwise go to step f.

e. Direct resolution test, second part.
Consider again the optimal solution (x_L^*, y_L^*) of LR.
Check if (x_L^*, y_L^*) is rational for the initial problem: solve FS(x_L^*), and let y_{FS}^* be an optimal solution.
If $d^2 y_L^* = d^2 y_{FS}^*$ then (x_L^*, y_L^*) is rational; otherwise go to step f.
Update z_{opt} and $(x_{\text{opt}}, z_{\text{opt}})$ if $z_{\text{opt}} < c^1 x_L^* + d^1 y_L^*$ and go to step m.

f. Second direct optimality test.
Compute all penalties p_i associated with strictly positive slack variables in the optimal tableau of LR. Set the other p_i equal to 0. Then for all k such that $r_k \in R$, compute

$$\pi_k = \min_{i \in I_k} p_i,$$

and set

$$\Pi = \max_k \pi_k.$$

If $z_{\text{opt}} \ge z_L^* - \Pi$, go to step m.

g. Second direct feasibility test.
If LR is infeasible, go to step m.

h. First conditional optimality test.
Consider again LR with the penalties p_i. For all i such that $z_{\text{opt}} \ge z_L^* - p_i$, fix α_i at 0 and update R.

i. Third direct optimality test.
If R contains a relation r_k such that $\alpha_j = 0$ for all $j \in I_k$, go to step m.

j. Relational optimality test.
For all remaining y_j appearing in $z_2(x, y)$, add to R the logical relations (14) or (15) on the α_i, if they are nonredundant. Eliminate from R those relations that have become redundant.

k. Second conditional optimality test.
If R contains a relation r_k such that $\alpha_j = 0$ for all $j \in I_k$ except for one index i, set the corresponding α_i to 1. Eliminate from the subproblem a variable y_j remaining in the ith constraint such that fill-in is minimum and return to step b.

l. Branching.
Apply the selected branching rule (the choice of which is discussed in §5) to choose either a free variable α_i or a relation $r_k \in R$ for which all variables α_i with $i \in I_k$ are free. In the former case, branch by fixing α_i at 1. In the latter case, branch by fixing the first variable α_i in r_k equal to 1. Eliminate a variable y_j remaining in the ith constraint such that fill-in is minimum. Return to step b.

m. Backtracking.
If branching took place on a variable, find the last α_i branched upon and equal to 1, set this $\alpha_i = 0$ and free the α_j fixed at 0 after α_i was fixed at 1. Otherwise consider the last logical relation r_k for which less than $|I_k|$ branches have been explored; consider the next branch. If there is no such variable or relation, stop. Otherwise update the current subproblem and return to step b.

Various heuristics may be used to obtain a first rational solution in the absence of first-level constraints. One of them, used in our implementation, consists of solving LR with the objective function $\lambda c^1 x + (1-\lambda) d^2 y$, i.e., a weighted sum of the leader's objective function for the x variables and of the follower's objective function for the y variables. The weight λ was chosen to be equal to

$$\frac{n_1 + n_2}{n_1 + 2n_2}.$$

Better heuristic solutions could be obtained at higher computational cost, e.g., by varying λ between 0 and 1 in the following objective function $(1-\lambda)(c^1x + d^1y) + \lambda d^2 y$ and keeping the first rational solution obtained as in Bard's algorithm [5], or by using Judice and Faustino's algorithm as they suggest in [28].

THEOREM 4.3. *Algorithm* HJS *solves* LBP *in a finite time.*

Proof. As mentioned in §2, LBP has an optimal solution at an extreme point of Ω, or has no solution. Branching on the tightness of the follower's constraints implies an implicit enumeration of all bases of FS, which are finite in number, as is the number of values for the vector α. Moreover, all steps take a finite time and the algorithm cannot cycle, as they are done in sequence unless simplification or branching takes place, which can happen only a finite number of times. □

As mentioned above, Algorithm HJS is for the tie cooperative case. Therein, ties in the values of the follower's objective function are automatically resolved as the leader wishes. This is ensured by LR: the objective function is that of the leader, so that if in tests d and e the solution (x_L^*, y_L^*) is found to be rational, it is the best from the leader's point of view among all rational solutions with $\overline{x} = x_L^*$. Algorithm HJS is easily adapted to solve the tie noncooperative case. Indeed, it suffices to add a secondary objective function in the follower's subproblem equal to $-d^1 y$ and activated only in case of ties for the objective function $d^2 y$, in a similar way as in preemptive goal programming (see, e.g., Ignizio [25]).

This modification, which affects tests d and e, has two consequences: first, rational solutions in case of ties for $d^2 y$ will usually differ from (x_L^*, y_L^*), so more branches will have to be explored; second, again in case of ties for $d^2 y$, the optimal solution may be modified.

5. An example. Consider the following two-level linear program (from Candler and Townsley [16], with an additional first-level constraint). The α_i are written in parentheses next to the corresponding constraints:

$$\max_x\ z_1 = 8x_1 + 4x_2 - 4y_1 + 40y_2 + 4y_3, \tag{25}$$

$$\text{subject to: } x_1 + 2x_2 - y_3 \le 1.3, \tag{26}$$

$$x_1 \ge 0, \quad x_2 \ge 0, \tag{27}$$

$$\max_y\ z_2 = -2y_1 - y_2 - 2y_3, \tag{28}$$

subject to:

$$(\alpha_1) \quad -y_1 + y_2 + y_3 \le 1, \tag{29}$$

$$(\alpha_2) \quad 4x_1 - 2y_1 + 4y_2 - y_3 \le 2, \tag{30}$$

$$(\alpha_3) \quad 4x_2 + 4y_1 - 2y_2 - y_3 \le 2, \tag{31}$$

$$(\alpha_4) \quad y_1 \ge 0, \tag{32}$$

$$(\alpha_5) \quad y_2 \ge 0, \tag{33}$$

$$(\alpha_6) \quad y_3 \ge 0. \tag{34}$$

Solve the problem:

$$\max_{x,y}\ z_R = \lambda c^1 x + (1-\lambda) d^2 y \quad \text{with } \lambda = \frac{n_1 + n_2}{n_1 + 2n_2} = \frac{5}{7},$$

i.e., $z_R = 5x_1 + 2.5x_2 - 1.5y_1 - 15y_2 + 1.5y_3$ subject to constraints (26)–(27) and (29)–(34) (step a). Its optimal solution $(x_R^*, y_R^*) = (1.5, 0, 1, 0, 2)$, with value $z_R^* = 9$, is feasible for problem (25)–(34). Therefore, $(x_{\text{opt}}, y_{\text{opt}})$ is initialized to (x_R^*, y_R^*) and z_{opt} to 16.

Solving the LR leads to $(x_L^*, y_L^*) = (0, 0, 1.5, 1.5, 1)$ and $z_L^* = 58 > z_{\text{opt}}$ (step b). As no constraint has yet been imposed on the dual variables of the follower's problem, it is feasible (step c). Setting $\overline{x}$ to x_L^*, we obtain $y_F^* = (0, 0, 0)$ and $z_2(y_F^*) = 0 \ne z_2(y_L^*) = -6.5$; hence (x_L^*, y_L^*) is not rational (step d).

The penalties p_i associated with constraints (29)–(34) are equal to:

$$p_1 = 0\,,\ p_2 = 0\,,\ p_3 = 0\,,\ p_4 = 28.8\,,\ p_5 = 42, \text{and } p_6 = 22 \qquad \text{(steps f and h)}.$$

The set R of logical conditions

$$\begin{aligned} \alpha_1 + \alpha_2 + \alpha_4 &\ge 1 \quad (\text{w.r.t. } y_1),\\ \alpha_3 + \alpha_5 &\ge 1 \quad (\text{w.r.t. } y_2),\\ \alpha_2 + \alpha_3 + \alpha_6 &\ge 1 \quad (\text{w.r.t. } y_3) \end{aligned}$$

are obtained (step j). We branch on the second relation and first consider $\alpha_5 = 1$ as $p_5 > p_3$. This leads to $y_2 = 0$ and the problem reduces to

$$\max_x z_1 = 8x_1 + 4x_2 - 4y_1 + 4y_3,$$

$$\text{subject to: } x_1 + 2x_2 - y_3 \le 1.3,$$

$$x_1 \ge 0, \qquad x_2 \ge 0;$$

$$\max_{y_1, y_3}\ z_2 = -2y_1 - 2y_3,$$

subject to:

$$\begin{aligned} -y_1 + y_3 &\le 1,\\ 4x_1 - 2y_1 - y_3 &\le 2,\\ 4x_2 + 4y_1 - y_3 &\le 2,\\ y_1 \ge 0, \qquad y_3 &\ge 0. \end{aligned}$$

The optimal solution (x_L^*, y_L^*) of LR is (1.5, 0, 1, 2) with a value $z_L^* = 16 \le z_{\text{opt}}$ (step b).

The incumbent solution remains: $z_{\text{opt}} = 16$, $(x_{\text{opt}}, y_{\text{opt}}) = (1.5, 0, 1, 0, 2)$.

Backtracking (step m) leads to the branch $\alpha_5 = 0$, $\alpha_3 = 1$. Variable y_1 is eliminated, leading to the subproblem

$$\begin{aligned} &\max_x z_1 = 8x_1 + 8x_2 + 38y_2 + 3y_3 - 2,\\ &\text{subject to: } x_1 + 2x_2 - y_3 \ge 1.3,\\ &\qquad x_1 \ge 0, \qquad x_2 \ge 0,\\ &\qquad \max_{y_2, y_3} z_2 = 2x_2 - 2y_2 - 2.5y_3 - 1,\\ &\qquad \text{subject to:}\\ &\qquad (\alpha_1) \qquad x_2 + 0.5y_2 + 0.75y_3 \le 1.5,\\ &\qquad (\alpha_2) \qquad 2x_1 + x_2 + 1.5y_2 - 0.75y_3 \le 1.5,\\ &\qquad (\alpha_4) \qquad x_2 - 0.5y_2 - 0.25y_3 \le 0.5,\\ &\qquad (\alpha_5) \qquad y_2 > 0,\\ &\qquad (\alpha_6) \qquad y_3 \ge 0. \end{aligned}$$

The current set of logical relations R (step j) is equal to $\{\alpha_4 \ge 1\}$ (with respect to y_2 in the above subproblem). Indeed the constraint $\alpha_2 + \alpha_4 + \alpha_6 \ge 1$, corresponding to y_3 in the current subproblem, is redundant and the three previous constraints of R are eliminated, two of them because they contain $\alpha_3 = 1$ and the third one because $\alpha_4 = 1$ makes it redundant. Tightness of the fourth constraint (step k), i.e., $\alpha_4 = 1$, allows us to eliminate y_2, leading to the subproblem

$$\begin{aligned} &\max_x z_1 = 8x_1 + 84x_2 - 16y_3 - 40,\\ &\text{subject to: } x_1 + 2x_2 - y_3 \le 1.3,\\ &\qquad x_1 \ge 0, \qquad x_2 \ge 0,\\ &\qquad \max_{y_3} z_2 = -2x_2 - 1.5y_3 + 1,\\ &\qquad \text{subject to:}\\ &\qquad (\alpha_1) \qquad 2x_2 + 0.5y_3 \le 2,\\ &\qquad (\alpha_2) \qquad 2x_1 + 4x_2 - 1.5y_3 \le 3,\\ &\qquad (\alpha_5) \qquad -2x_2 + 0.5y_3 \le -1,\\ &\qquad (\alpha_6) \qquad y_3 \ge 0. \end{aligned}$$

A new logical relation is found : $\alpha_2 + \alpha_6 \ge 1$. Computing the optimal solution (x_L^*, y_L^*) of LR, its value z_L^* and the penalties p_i yield $(x^*, y_L^*) = (0, .833, .466)$, $z_L^* = 26.73 > z_{\text{opt}}$, and $p_1 = 0$, $p_2 = 8.333$, $p_6 = 12.333$. Setting $\overline{x}$ to x_L^* in FR yields the solution $y_{FR}^* = (.3555)$ and $z_2(y_F^*) = -1.3 \ne z_2(y_L) = -1.466$; hence (x_L^*, y_L^*) is not rational.

Penalty $p_6 = 12.33$ is such that $z_{\text{opt}} > z_L^* - p_6$; hence $\alpha_6 = 0$ (step h). The set R reduces to $\{\alpha_2 \ge 1\}$. Tightness of the second constraint allows us to eliminate y_3, the last of the follower's variables. This gives the following subproblem

$$\begin{aligned}
\max_x z_1 &= -13.33x_1 + 41.33x_2 - 8\\
\text{subject to: } & -x_1 - 2x_2 \le -2.1\\
& x_1 \ge 0, \quad x_2 \ge 0;\\
& \max_y z_2 = -2x_1 - 6x_2 + 4\\
& \quad \text{subject to:}\\
& \quad (\alpha_1) \quad 2x_1 + 10x_2 \le 9,\\
& \quad (\alpha_5) \quad x_1 - x_2 \le 0,\\
& \quad (\alpha_6) \quad -2x_1 - 4x_2 \le -3.
\end{aligned}$$

Solving LR yields $x_L^* = (0.5, 0.8)$ and $z_L^* = 18.4$, which is trivially rational for the current subproblem. Moreover, setting $\overline{x}$ to x_L^* in FS yields the solution $y_{FS}^* = (0, .2, .8)$ with $z_2(y_{FS}^*) = z_2(y_L^*) = -1.8$. Thus the solution (0.5, 0.8, 0, 2, 0.8) is rational for the initial problem, and the incumbent vector and value are updated. Backtracking brings the algorithm to stop. The problem is thus solved after examining three nodes only (see Fig. 1).

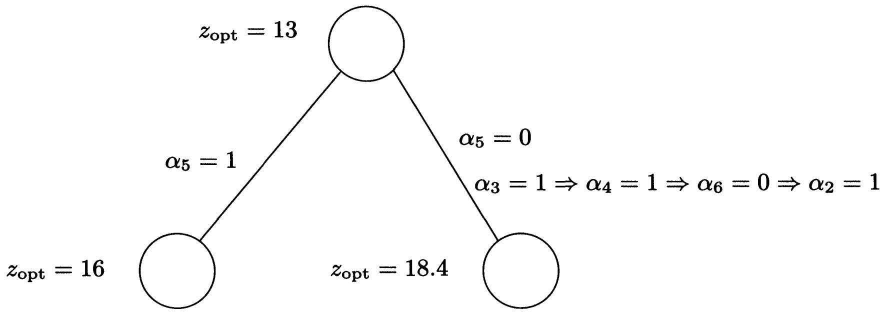

FIG. 1. *Branch-and-bound tree for the example.*

6. Computational experience.

The algorithm of §4 has been coded in FORTRAN 77 and extensively tested on a SUN 3/50 and, for one series of tests, on a SUN SPARC computer. We also solved some test problems with the program of Bard and Moore [9] (referred to as BM below) on the same computer. Algorithm BM is similar to that of Fortuny-Amat and McCarl [19] in that both use branching on the complementary slackness conditions associated with the follower's subproblem. Fortuny-Amat and McCarl convert the complementary conditions into mixed-integer linear constraints before branching. The relaxation of the problem so obtained is less tight than in the BM algorithm. Bard and Moore consider the relaxation of LBP obtained by deleting the follower's objective function (i.e., program LR), then add the constraints of the dual of the follower's subproblem, solve in both the primal and dual variables, and branch on the complementary slackness conditions until they are satisfied. The program BM uses Marsten's code XMP [31] as a subroutine to solve the linear programs. We used the same code for that purpose.

TABLE 1
Comparison of branching criteria.

	n_1	42		42		42		42		60		50		70		60		70	
	n_2	28		28		28		28		40		50		50		60		60	
	m_1	0		0		0		7		0		0		0		0		0	
	m_2	28		28		28		21		40		40		48		48		52	
	d	40%		17%		8%		8%		8%		8%		8%		8%		8%	
		nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu
BR1	μ	4959.8	7830.4	416.6	250.3	37.5	18.7	3433.2	1754.6	1182.6	1045.1	2772.1	3984.5	4800.5	9098.4	—	—	—	—
	σ	4479.2	7576.9	1244.8	727.5	78.5	29.2	7823.8	3841.3	3089.3	2603.2	4046.2	6095.0	4293.5	7623.2	—	—	—	—
	m	2439	1922.9	3	6.7	9	7.7	122	90.0	11	27.4	191	480.5	5150	15000.0	—	—	—	—
BR2	μ	4290.8	6380.2	377.6	227.7	37.3	18.0	3210.2	1669.8	261.7	284.7	1631.4	2288.1	4518.0	9109.6	—	—	—	—
	σ	4675.4	7464.1	1132.0	663.4	78.6	29.7	7761.4	3953.9	481.3	463.8	3046.3	4602.9	4095.8	7611.4	—	—	—	—
	m	733	635.7	3	6.4	7	5.6	29	27.6	9	20.3	115	206.3	5113	15000.4	—	—	—	—
BR3	μ	2917.4	4575.7	649.0	653.5	917.8	402.1	3699.9	3111.0	752.8	925.4	2512.2	4367.0	3722.8	8478.0	—	—	—	—
	σ	4574.6	7194.8	1565.9	1595.5	1910.2	826.9	5820.0	4801.3	881.2	1144.7	2966.4	5428.2	3109.0	7175.6	—	—	—	—
	m	73	100.5	7	13.0	13	11.23	235	172.9	215	325.5	1081	2226.5	3715.0	7352.3	—	—	—	—
BR4	μ	34.6	50.1	6.0	8.7	5.2	4.7	13.4	15.5	6.2	17.6	56.4	138.0	30.0	113.2	29.4	157.4	65.4	341.2
	σ	48.4	66.0	7.0	7.5	3.5	2.3	18.9	17.2	3.5	9.5	129.3	257.0	32.2	103.4	45.2	212.1	112.8	520.0
	m	9	15.3	3	6.5	7	5.5	5	8.0	5	14.4	13	44.4	19	69.3	11	71.4	11	68.7
BR5	μ	32.6	47.5	5.4	7.9	5.4	4.6	10.8	13.0	6.4	18.0	17.6	56.8	30.4	112.3	24.0	126.0	26.2	154.8
	σ	44.1	59.9	5.2	5.7	3.8	2.4	10.2	9.8	3.6	9.4	12.8	35.7	31.3	98.5	29.4	126.6	26.2	140.8
	m	9	15.1	3	6.4	7	5.3	5	8.0	5	14.2	11	41.1	19	83.4	11	69.5	11	68.7
BR6	μ	72.2	96.7	5.6	8.2	5.4	4.6	14.4	15.8	7.6	20.6	22.6	67.0	34.2	121.3	36.4	174.6	22.6	133.7
	σ	108.3	132.7	5.3	5.7	3.8	2.3	19.1	17.6	5.0	13.0	14.9	42.8	29.4	95.6	50.6	205.9	20.3	109.4
	m	9	14.9	3	6.4	7	5.3	5	8.0	5	14.1	21	59.0	25	85.2	15	76.8	11	70.8
BR7	μ	1065.4	1317.1	6.0	8.6	6.0	5.0	76.8	73.2	8.0	20.6	28.2	76.1	56.8	193.3	46.2	206.9	33.6	185.3
	σ	1660.8	2094.3	6.5	6.9	4.5	2.5	160.9	151.6	5.7	13.0	22.2	53.5	66.3	200.6	62.4	253.1	39.9	201.5
	m	13	19.7	3	6.4	7	5.6	13	13.8	5	14.1	17	45.7	21	84.3	27	117.5	19	120.9

Problems were generated randomly with the same characteristics as in Bard and Moore [9] for 40 percent and 17 percent density and as in Savard [35] for sparser (8 percent and 6 percent density) ones. To avoid empty columns, coefficients were generated column after column with these densities. To avoid unbounded optimal solutions in the follower's subproblem if all coefficients in a column corresponding to a variable y_j with $d_j^2 > 0$ are negative, the first one is multiplied by -1; then boundedness is checked and unbounded problems deleted.

The main parameters considered are the numbers n_1 and n_2 of leader and follower variables, m_1 and m_2 of first-level and second-level constraints and density d of the coefficient matrix. For each set of these values in each series of tests, 10 problems are solved. Mean (μ) and median (m) values as well as standard deviations (σ) for the number of nodes in the branch-and-bound tree and for the cpu times are given.

A first class of experiments was aimed at streamlining the algorithm and assessing which of its features are the most useful. Numerous branching rules were tested. In Table 1 we present results for the seven best rules, which are described next. We denote by s_i the slack variable associated with constraint i of type (5) or (6) and by u_i the corresponding dual variable for $i = 1, 2, \cdots, m_2 + n_2$.

BR1. *Multiple branching.* (i) Select the logical relation $r_k \in R$ with smallest cardinality. (ii) Break ties by choosing a relation with the largest number of slack variables s_i in basis. (iii) Order the variables α_i in r_k by decreasing values of the penalties p_i.

BR2. Same as BR1, except for (iii). Order the variables α_i by decreasing values of the products $s_i u_i$.

BR3. *Binary branching.* Select the boolean variable α_i associated with the largest penalty p_i.

BR4. Select the α_i associated with the largest product $s_i u_i$ (same rule as in Algorithm BM).

BR5. *Multiple or binary branching.* (i) Select the relation $r_k \in R$ with two variables α_i and both the corresponding slack variables s_i in basis, such that $\sum_{i \in I_k} u_i s_i$ is maximum. (ii) If there is no such relation, use BR4.

BR6. Same as BR5 except that in (ii), branch on the variable α_i with largest penalty p_i among those for which $s_i u_i > 0$.

BR7. Same as BR5 except that in (ii), branch on the variable with smallest product $s_i u_i > 0$.

It appears that:

(i) numbers of nodes generated and computation times are extremely sensitive to the chosen branching rule;

(ii) multiple branching on logical relations involving few α_i is less efficient than dichotomous branching on the α_i;

(iii) best results are obtained by a hybrid rule in which multiple branching is first used and one switches to dichotomous branching when no more logical relations with few α_i are available;

(iv) with all branching rules, difficulty of resolution varies greatly from problem to problem: often $\sigma > m$ and $m << \mu$. Many problems are easy to solve, with one or a few nodes in the branch-and-bound tree, while a few take several thousand nodes.

As branching rules BR5 and BR6 gave the best results, they were adopted in the remaining experiments.

In a second series of tests the effect of heuristics was assessed by obtaining an upper bound on the reduction in computing effort they can provide in the following

TABLE. 2a
No heuristic.

	n_1	50		70		60		70	
	n_2	50		50		60		60	
	m_2	40		48		48		52	
	d	8%		8%		8%		8%	
		nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu
BM	μ	2863.2	2504.8	2111.7	3628.3	1981.6	4238.4	3658.3	7302.3
	σ	4968.3	4569.9	3114.2	4999.1	2844.7	5853.9	3462.3	6332.3
	m	396	330.2	518	1208.2	662	1420.5	2138.0	4040.9
BR5	μ	18.0	59.4	30.4	112.6	24.6	132.4	26.2	159.3
	σ	12.6	37.7	31.3	98.3	29.0	129.5	26.2	144.8
	m	13	53.2	19	83.3	11	69.0	11	73.2
BR6	μ	23.2	70.8	34.2	123.3	37.0	181.6	22.6	136.1
	σ	15.0	47.1	29.4	97.4	50.2	205.8	20.3	111.5
	m	21	63.7	25	85.6	15	77.8	11	75.0

TABLE 2b
HJS *heuristic.*

	n_1	50		70		60		70	
	n_2	50		50		60		60	
	m_2	40		48		48		52	
	d	8%		8%		8%		8%	
		nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu
BM	μ	2175.0	1991.2	1959.2	3421.9	1953.7	4209.6	3357.4	7303.1
	σ	4828.4	4525.3	3114.7	5010.3	2859.3	5872.3	2989.2	6332.5
	m	342	250.3	518	1205.6	466	1153.5	2138	4040.6
BR5	μ	17.6	56.8	30.4	112.3	24.0	126.0	26.2	154.8
	σ	12.8	35.7	31.3	98.5	29.4	126.6	26.2	140.8
	m	11	41.1	19	83.4	11	69.5	11	68.7
BR6	μ	22.6	67.0	34.2	121.3	36.4	174.6	22.6	133.7
	σ	14.9	42.8	29.4	95.6	50.6	205.9	20.3	109.4
	m	21	59.0	25	85.2	15	76.8	11	70.8

TABLE 2c
Using the optimal solution as first heuristic solution.

	n_1	50		70		60		70	
	n_2	50		50		60		60	
	m_2	40		48		48		52	
	d	8%		8%		8%		8%	
		nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu
BM	μ	876.0	833.6	1299.7	2700.1	1722.5	3719.7	2032.6	4675.5
	σ	1935.6	1822.3	2190.6	4574.7	2914.1	6011.2	2452.6	5643.5
	m	138	98.0	304.0	598.7	160	652.1	944	2068.6
BR5	μ	12.4	42.7	15.2	64.9	18.8	99.2	17.0	97.7
	σ	11.2	30.8	14.3	46.8	30.1	129.7	21.0	97.1
	m	7	27.6	9	424	3	31.1	9	57.7
BR6	μ	16.4	51.5	17.4	70.6	25.0	119.1	13.8	85.4
	σ	13.8	39.4	16.5	53.7	48.6	196.5	11.5	66.5
	m	11	38.2	9	42.4	5	33.9	9	57.6

way: a series of test problems solved previously was solved again, first using no heuristic and then assuming the optimal value to be known a priori. Results are given in Tables 2a, 2b, and 2c (lines corresponding to BM will be discussed later) and are to be compared with those of Table 1. It appears that:

(i) a priori knowledge of the best solution significantly decreases computational effort (average cpu times are reduced by 21.3–42.2 percent);

TABLE 3
Influence of penalties.

	n_1	42				42				42				60			
	n_2	28				28				28				40			
	m_2	28				28				28				40			
	d	40%				17%				8%				8%			
				fixations				fixations				fixations				fixations	
		nodes	cpu	rlog	pen	nodes	cpu	rlog	pen	nodes	cpu	rlog	pen	nodes	cpu	rlog	pen
BR6	μ	68.8	60.6	1.5	—	7.2	7.0	5.4	—	7.6	4.2	15.9	—	11.4	15.9	12.4	—
	σ	87.6	76.1	1.7	—	7.3	5.0	2.7	—	5.8	2.2	5.0	—	8.3	8.6	4.0	—
	m	29	24.1	1.0	—	5	5.7	5.0	—	7	4.1	15.0	—	7.0	10.9	12.0	—
BR6	μ	33.0	43.0	4.3	30.5	5.4	7.4	6.4	6.7	5.4	4.4	16.9	7.6	6.4	16.9	21.5	23.3
	σ	44.9	54.6	5.3	30.0	5.2	5.2	2.7	8.3	3.9	2.1	5.6	6.9	3.7	8.5	9.5	24.4
	m	9	13.7	3.0	20.0	3	6.1	7.0	3	7	5.0	14.0	8.0	5.0	13.3	20.0	14

	n_1	50				70				60				70			
	n_2	50				50				60				60			
	m_2	40				48				48				52			
	d	8%				8%				8%				8%			
				fixations				fixations				fixations				fixations	
		nodes	cpu	rlog	pen	nodes	cpu	rlog	pen	nodes	cpu	rlog	pen	nodes	cpu	rlog	pen
BR6	μ	41.8	64.4	30.4	—	53.2	102.8	22.3	—	44.2	126.6	19.0	—	82.6	268.7	24.9	—
	σ	37.5	48.7	16.4	—	50.2	85.7	12.9	—	48.2	124.2	7.2	—	105.9	331.4	19.4	—
	m	23	37.8	19.0	—	41	83.9	22.0	—	23	66.2	16.0	—	21	78.5	14	—
BR6	μ	17.6	53.6	51.3	49.6	30.4	103.9	48.0	75.9	24.0	119.0	40.3	66.2	26.2	145.7	36.7	57.10
	σ	12.8	33.3	25.8	38.0	31.3	90.4	33.1	57.6	29.4	119.2	28.6	56.0	26.2	132.0	25.1	55.21
	m	11	38.7	46.0	42.0	19	77.5	43.0	56.0	11	66.5	26.0	50.0	11.0	65.4	30.0	30.00

(ii) some problems remain difficult to solve;

(iii) the heuristic described in §5 only entails small reductions in numbers of nodes and cpu time.

So, there is room for improved heuristics (e.g., as discussed above, those of Bard [5] and Judice and Faustino [27], [28]) but heuristics alone will not always render the solution of LBP easy.

In the next series of tests the effect of penalties was assessed by solving the same series of problems without and with them. Results, given in Table 3, show that:

(i) using penalties significantly reduces the number of nodes in the branch-and-bound tree (the average reduction in number of nodes is between 25.0 and 68.3 percent);

(ii) cpu times are not much affected (sometimes slightly increased and sometimes slightly decreased) except for the largest test problems, in which case using penalties reduces mean cpu time by 45.8 percent.

Table 3 also gives statistics on the number of times a variable α_i is fixed using a logical relation (rlog) and using a test involving penalties (pen). Note that the number of fixations due to logical relations increases when penalties are used, as these penalties can show that some α_i must be equal to 0 and hence sharpen the logical relations in which these α_i appear.

A second class of experiments aims at studying which parameters influence the difficulty of resolution of LBP. Results of a sensitivity analysis on the right-hand side vector are given in Table 4. Each component of this vector is chosen to be equal to s times the fraction of the right-hand side minus the sum of the negative coefficients over the sum of the absolute values of the coefficients in the corresponding constraint; s is varied from 0.4 to 0.8. No clear-cut tendency appears to be detectable. Contrasting with this result, very large differences in problem difficulty are observed when the amount by which the leader's and follower's objective functions are antagonistic varies (this amount depends on the ranges and the signs of the coefficients in both objective functions). Table 5 shows results obtained when coefficients d_j^1 in the leader's objective function and coefficients d_j^2 in the follower's objective function are randomly chosen in a uniform distribution on $[\ell, 20]$ where ℓ varies from -20 to $+12$. It appears that:

(i) computational effort regularly decreases when the lower bound ℓ increases, i.e., when the leader's and follower's objective functions become closer to being parallel.

(ii) computational difficulty of the hardest and easiest problems is very different (averages 33.3–56.3 times more nodes and 12.3–19.6 times more cpu time in the former than in the latter case).

This suggests that in reporting future experiments on algorithms for LBP, a parameter such as ℓ specifying the amount of antagonism between the two objective functions should be systematically specified.

In another series of experiments the number of first-level variables was varied while the numbers of second-level variables and of constraints were kept constant. Results of Table 6 show that computational effort significantly *decreases* when the number of first-level variables *increases*. It thus appears that the number of follower's variables is not in itself a sufficient measure of computational difficulty for LBP.

A third class of experiments was aimed at comparing our algorithm with the best previous ones, those of Bard and Moore [9] and of Judice and Faustino [27], [28]. A direct comparison with the algorithm of Bard and Moore was possible, as they kindly made their code available to us. Results are reported in Table 2 and Tables 7a and

TABLE 4
Sensitivity analysis on the right-hand side vector.

		$n_1=50$, $n_2=50$, $m_1=0$, $m_2=40$, $d=8\%$						$n_1=60$, $n_2=60$, $m_1=0$, $m_2=48$, $d=8\%$					
		$s=.4$		$s=.6$		$s=.8$		$s=.4$		$s=.6$		$s=.8$	
		nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu
BR5	μ	53.4	137.3	47.0	111.7	64.2	161.3	188.2	706.6	213.0	787.7	196.6	754.4
	σ	76.0	156.9	80.9	147.7	111.8	258.3	471.4	1695.4	447.3	1534.3	336.4	1227.1
	m	19	61.9	19.0	57.1	19	58.5	19	101.4	55	284.3	45	231.0
BR6	μ	44.5	124.6	4.0	108.7	62.3	152.2	249.1	868.6	220.2	790.3	297.4	1047.4
	σ	45.9	118.8	52.8	116.4	99.4	249.0	466.2	1443.4	324.9	1016.5	533.4	1617.5
	m	19	75.3	19	57.5	19	75.7	43	196.0	95	413.8	43	212.1

TABLE 5
Sensitivity analysis on the coefficients of the objective function.

	$n_1=50$, $n_2=50$, $m_1=0$, $m_2=40$, $d=8\%$	$c\in[-20,20]$		$c\in[-12,20]$		$c\in[-4,20]$		$c\in[0,20]$		$c\in[4,20]$		$c\in[12,20]$	
		nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu
BR5	μ	63.3	149.0	25.1	66.7	13.4	40.0	3.8	16.4	3.3	15.2	1.9	12.1
	σ	63.6	135.9	29.9	71.1	29.2	69.9	5.5	14.6	4.1	12.8	1.5	4.8
	m	29	82.5	9	27.5	5	20.2	3	14.6	1	10.3	1	9.7
BR6	μ	106.5	237.3	24.1	65.1	10.3	33.1	4.2	17.2	2.9	13.9	1.9	12.1
	σ	155.1	329.2	28.3	69.2	19.9	48.6	7.0	17.7	2.8	8.7	1.5	4.8
	m	31	90.9	9	33.2	5	20.1	3	14.5	1	10.2	1	9.6

TABLE 6
Increasing the number of first-level variables.

	n_1	50		75		100		150		200		250		300		350	
	n_2	60		60		60		60		60		60		60		60	
	m_2	100		100		100		100		100		100		100		100	
	d	8%		8%		8%		8%		8%		8%		8%		8%	
		nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu
BR5	μ	619.8	908.5	64.8	126.9	141.0	286.7	12.0	35.5	20.2	86.5	9.0	37.1	7.8	38.0	4.0	27.5
	σ	966.4	1377.1	83.9	156.2	232.1	466.9	10.1	26.7	23.6	131.1	15.1	48.6	7.4	27.6	4.2	18.4
	m	221	333.6	15	30.4	49	112.9	7	25.6	13	51.7	3	21.9	5	25.1	1	19.0

7b. It appears that:

(i) Algorithm HJS with branching rule BR5 always outperforms Algorithm BM and does the same with branching rule BR6 with two exceptions;

(ii) The ratios of computation times for Algorithms BM and HJS with branching rule BR5 are between 1.4 and 9.6 when $d = 40$ percent, 3.5 and 49.0 when $d = 17$ percent, 23 and 47.1 when $d = 8$ percent, but vary widely in each of these cases;

(iii) This ratio increases with problem size;

(iv) Similar conclusions hold even if no heuristic is used or if the optimum value of LBP is assumed to be known *a priori* (see Table 2).

Reasons for which Algorithm HJS is faster than Algorithm BM appear to be the following (probably in decreasing order of importance):

(i) use of logical relations (14) and (15); for sparse problems some of these logical relations may have only one α_i and thus lead to problem simplification and possibly to complete solution without branching;

TABLE 7a
Comparison of HJS *and* BM *algorithms on problems without first-level constraints.*

	n_1	20		50		42		35		60		45	
	n_2	30		20		28		35		30		45	
	m_2	20		28		28		28		36		36	
		nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu
40%	μ	175.2	130.8	67.6	148.2	158.4	306.9	579.0	1080.7	203.0	932.6	766.2	2957.5
BM	σ	239.0	169.8	60.6	116.7	199.3	337.6	1244.5	2219.2	165.6	719.3	877.7	3469.5
	m	68	59.2	30	75.0	82	172.2	146	306.2	148	733.1	364	1259.8
40%	μ	105.8	92.4	10.2	13.0	32.6	47.5	211.2	339.7	60.2	136.2	441.8	1292.3
BR5	σ	195.8	159.0	20.5	19.3	44.1	59.9	449.4	692.2	58.2	127.2	520.2	1513.0
	m	5	9.6	1	3.6	9	15.1	41	80.5	43	92.7	81	244.3
40%	μ	271.2	223.7	14.2	16.7	72.2	96.7	398.6	614.9	110.2	235.4	1637.9	4662.4
BR6	σ	638.5	506.6	31.6	29.1	108.3	132.7	659.8	988.9	116.3	243.5	2274.0	6486.7
	m	5	9.6	1	3.6	9	14.9	71	122.6	61	129.7	117	345.5
17%	μ	162.4	46.7	34.6	42.7	180.0	192.8	753.8	788.6	86.0	249.7	551.8	1097.9
BM	σ	165.2	45.2	24.2	22.4	252.7	265.0	1079.8	1057.2	137.4	350.6	1459.9	2659.3
	m	28	12.3	20	28.2	32	39.5	252	275.7	36	116.8	80	267.2
17%	μ	18.2	13.3	8.6	12.2	5.9	7.9	81.4	104.5	4.8	12.4	20.8	58.2
BR5	σ	21.3	14.4	10.1	12.9	5.2	5.7	96.2	114.7	7.6	13.4	32.1	79.0
	m	5	5.9	3	5.2	3	6.4	17	30.4	1	5.9	5	17.5
17%	μ	24.6	16.4	4.2	4.9	5.6	8.2	53.8	74.4	6.6	15.2	27.6	70.8
BR6	σ	34.4	20.9	3.9	3.3	5.3	5.7	62.6	80.4	12.4	20.8	51.3	114.6
	m	5	5.9	3	3.8	3	6.4	15	26.7	1	5.9	5	17.5
8%	μ							76.6	14.4	58.0	40.6	2290.0	1317.2
BM	σ							98.2	14.4	60.6	44.2	4522.5	2511.0
	m							40	9.6	24	20.0	600	359.1
8%	μ							5.2	6.4	6.0	9.0	30.4	58.6
BR5	σ							4.5	4.0	5.8	6.5	42.1	65.2
	m							3	4.4	3	5.9	19	45.4
8%	μ							5.4	6.3	6.0	8.6	17.6	38.4
BR6	σ							4.9	4.0	5.7	6.2	15.3	27.9
	m							3	4.4	3	6.0	9	27.0

(ii) solution of LR, and the dual of FR separately instead of jointly, thus reducing the size of the LPs solved in various tests (Algorithm BM includes the dual variables in the master program and hence contains m_2 more variables than does Algorithm HJS);

(iii) use of hybrid branching rules;

(iv) use of penalties.

In Table 8 we consider problems similar to those considered in the previous series of experiments but transfer some of the second-level constraints to the first level. These problems are solved by Algorithm HJS and a version of Algorithm BM, slightly modified to allow for first-level constraints with y variables. It appears that:

(i) computational difficulty sometimes increases and sometimes decreases when constraints are transferred from second to first level;

(ii) increase in computing time when there are first-level constraints does not exceed a factor of 3 with Algorithm HJS;

(iii) Algorithm HJS always outperforms Algorithm BM, and the ratios of computing times are higher when there are first-level constraints (from 2.83 to 53.3) than when there are none.

Comparison of Algorithm HJS with Algorithm JF of Judice and Faustino [27], [28] is more difficult, as a code for the latter is not generally available at this time.

Judice and Faustino [28] present computational results for several large sparse

TABLE 7b
Comparison of HJS *and* BM *algorithms on problems without first-level constraints.*

	n_1	70		60		50		70		60		70	
	n_2	30		40		50		50		60		60	
	m_2	40		40		40		48		48		52	
		nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu
40%	μ	77.4	519.5	128.8	434.4								
BM	σ	53.9	342.5	97.2	293.8								
	m	54	383.0	92	302.2								
40%	μ	17.2	54.2	48.6	160.8								
BR5	σ	22.4	55.8	72.7	215.0								
	m	3	29.3	7	32.2								
40%	μ	18.0	60.3	97.8	315.5								
BR6	σ	30.8	96.4	134.5	422.4								
	m	7	29.3	33	97.5								
17%	μ	260.6	926.2	1032.6	1739.2								
BM	σ	506.8	1559.2	2497.9	3920.4								
	m	62	290.8	44	119.6								
17%	μ	9.4	25.1	10.0	35.5								
BR5	σ	10.4	21.3	11.9	29.5								
	m	5	14.1	1	12.9								
17%	μ	9.6	25.7	8.4	30.8								
BR6	σ	9.7	20.0	11.5	28.7								
	m	5	16.6	1	12.8								
8%	μ	77.4	84.7	211.4	153.4	2863.2	2496.7	2114.0	3622.9	1984.7	4235.4	3663.9	7294.5
BM	σ	62.8	67.2	147.7	118.7	4968.3	4559.3	3119.9	4997.7	2850.9	5855.2	3469.5	6333.0
	m	46	50.0	168	100.1	396	329.2	518	1204.5	662	1417.1	2138	4022.6
8%	μ	5.8	10.8	6.4	18.0	17.6	56.8	30.4	112.3	24.0	126.0	26.2	154.8
BR5	σ	5.3	9.3	3.6	9.4	12.8	35.7	31.3	98.5	29.4	126.6	26.2	140.8
	m	3	6.1	5	14.2	11	41.1	19	83.4	11	69.5	11	68.7
8%	μ	5.2	9.5	7.6	20.6	22.6	67.0	34.2	121.3	36.4	174.6	22.6	133.7
BR6	σ	4.0	6.7	5.0	13.0	14.9	42.8	29.4	95.6	50.6	205.9	20.3	109.4
	m	3	6.1	5	14.1	21	59.0	25	85.2	15	76.8	11	70.8

problems with up to 150 constraints, 250 variables controlled by the leader, and 150 variables controlled by the follower, on a CYBER 180-830 computer. These problems belong to two classes: "nonconflicting" ones in which all coefficients in the leader's and follower's objective functions are positive, and "conflicting" ones in which a few coefficients in these objective functions differ in sign. We have generated similar problems with all coefficients of the objective functions taken randomly in $[0, 20]$ in the first case: 80 percent of them are taken in that way and 20 percent are taken in $[-20, 0]$ in the second case. Results obtained on a SUN SPARC computer are presented in Table 9 (which also contains results for a few slightly denser problems). It appears that computation times are roughly similar to those of JF, but the latter have been obtained on a much more powerful computer (CYBER) and with a tolerance of 1 or 2 percent of the optimal value, which is obtained by Algorithm HJS.

Acknowledgment. The authors are grateful to Jonathan F. Bard for making available to them the code for linear bilevel programming he recently wrote with J. T. Moore. They also thank Alain Haurie for several fruitful discussions of preliminary drafts, and one anonymous referee for his comments.

TABLE 8
Comparison of HJS *and* BM *algorithms on problems without first-level constraints.*

	n_1	35		60		45		70		60		50		70		60		70	
	n_2	35		30		45		30		40		50		50		60		60	
	m_1	7		9		9		10		10		10		12		12		13	
	m_2	21		27		27		30		30		30		36		36		39	
	d	8%		8%		8%		8%		8%		8%		8%		8%		8%	
		nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu
BM	μ	108.8	20.7	211.4	119.5	572.8	340.4	392.0	511.0	745.4	834.2	5116.5	2810.1	3033.6	5846.5	4719.0	7322.5	3862.8	8393.2
	σ	103.2	20.4	362.9	196.6	646.0	415.4	1042.9	1413.1	1097.4	1391.7	9031.6	4399.7	2909.9	5755.2	4244.3	6210.9	3258.5	5813.2
	m	72	14.4	36	27.8	310	201.1	56	54.2	280	392.2	2162	1499.3	1784	3169.5	2958	5114.7	2036	5000.9
BR5	μ	8.2	7.3	17.4	19.8	24.6	40.7	8.6	14.2	16.0	30.4	95.6	197.1	33.8	109.6	369.2	1133.7	301.2	1070.9
	σ	4.8	3.4	26.3	27.2	22.1	32.6	8.3	11.4	11.6	17.6	71.3	139.4	22.1	64.8	579.5	1669.9	447.7	1329.9
	m	7	6.2	5	7.7	17	30.3	5	9.2	11	22.9	83	171.9	27	81.1	57	240.4	101	448.4
BR6	μ	7.0	6.8	13.4	15.1	24.6	42.5	9.0	14.8	17.4	32.5	135.6	273.0	59.0	174.4	332.6	1052.0	285.4	1082.5
	σ	4.6	3.5	18.5	18.3	22.6	36.7	9.8	13.5	13.2	21.5	118.0	231.1	50.0	137.8	386.3	1226.1	306.7	1107.9
	m	5	6.2	5	7.7	17	34.1	5	9.3	9	20.1	71	142.2	27	78.6	61	218.8	173	657.6

TABLE 9
Results for some larger problems (SUN SPARC).

	n_1	70		80		80*		300**		250**		300***		250***	
	n_2	70		70		80		100		150		100		150	
	m_2	56		60		64		100		150		100		150	
	d	8%		8%		8%		6%		6%		6%		6%	
		nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu	nodes	cpu
BR5	μ	133.4	195.8	161.4	262.1	1379.1	2420.5	2.6	37.4	5.8	238.9	5.0	49.9	31.8	698.8
	σ	147.2	211.6	225.6	336.9	2866.0	4834.0	3.5	28.6	7.8	104.7	3.53	26.8	65.9	1052.9
	m	85	120.6	55	126.2	117.0	256.9	1	27.2	1	184.9	5	38.0	5	259.2

* one problem was not solved within 15000 seconds.

** $c \in [0, 20]$

*** 20 percent of the variables have conflicting coefficients, i.e., for 80 percent of them $(c^1, d^1), d^2 \in [0, 20]$, and for 20 percent of them, $(c^1, d^1) \in [0, 20], d^2 \in [-20, 0]$.

REFERENCES

[1] E. Aiyoshi and K. Shimizu, *Hierarchical decentralized systems and its new solution by a barrier method*, IEEE Trans. Systems Man Cybernet., SMC–11 (1981), pp. 444–449.

[2] ———, *A solution method for the static constrained Stackelberg problem via penalty method*, IEEE Trans. Automat. Control, AC–29 (1984), pp. 1111–1114.

[3] F. A. Al-Khayyal, R. Horst, and P. M. Pardalos, *Global optimization of concave functions subject to quadratic constraints: An application in nonlinear bilevel programming*, Ann. Oper. Res., to appear.

[4] G. Anandalingam and D. J. White, *A penalty function approach for solving bilevel linear programs*, Res. Report, Department of Systems Engineering, University of Pennsylvania, Philadelphia, November 1989.

[5] J. F. Bard, *An efficient point algorithm for a linear two-stage optimization problem*, Oper. Res., 31 (1983), pp. 670–684.

[6] ———, *An algorithm for solving the general bilevel programming problem*, Math. Oper. Res., 8 (1983), pp. 260–272.

[7] ———, *Convex two-level programming*, Math. Programming, 40 (1988), pp. 15–28.

[8] J. F. Bard and J. E. Falk, *An explicit solution to the multi-level programming problem*, Comput. Oper. Res., 9 (1982), pp. 77–100.

[9] J. F. Bard and J. T. Moore, *A branch and bound algorithm for the bilevel programming problem*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 281–292.

[10] W. F. Bialas and M. H. Karwan, *On two-level linear optimization*, IEEE Trans. Automat. Control, AC-27 (1982), pp. 211–214.

[11] ———, *Two-level linear programming*, Management Sci., 30 (1984), pp. 1004–1020.

[12] O. Ben-Ayed and C. E. Blair, *Computational difficulties of bilevel linear programming*, Oper. Res., 38 (1990), pp. 556–559.

[13] J. Bracken and J. McGill, *Mathematical programs with optimization problems in the constraints*, Oper. Res., 21 (1973), pp. 37–44.

[14] W. Candler, J. Fortuny-Amat, and B. McCarl, *The potential role of multilevel programming in agricultural economics*, Amer. J. Agricultural Econ., 63 (1981), pp. 521–531.

[15] W. Candler and R. Norton, *Multilevel programming and development policy*, No. 258, IBRD, World Bank Staff Working Paper, Washington DC, 1977.

[16] W. Candler and R. Townsley, *A linear two-level programming problem*, Comput. Oper. Res., 9 (1982), pp. 59–76.

[17] S. Dempe, *A simple algorithm for the linear bilevel programming problem*, Optimization, 18 (1987), pp. 373–385.

[18] J. E. Falk, *A linear max-min problem*, Math. Programming, 5 (1973), pp. 169–188.

[19] J. Fortuny-Amat and B. McCarl, *A representation and economic interpretation of a two-level programming problem*, J. Oper. Res. Soc., 32 (1981), pp. 783–792.

[20] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of* NP-*Completeness*, W.H. Freeman, San Francisco, 1979.

[21] P. Hansen, B. Jaumard, and S.-H. Lu, *A framework for algorithms in globally optimal design*, J. Mech. Trans. Automat. Design, 111 (1989), pp. 353–360.

[22] ———, *An analytical approach to global optimization*, Math. Programming, 52 (1991), pp. 227–254.

[23] A. Haurie, R. Loulou, and G. Savard, *A two-player model of power cogeneration in New England*, IEEE Trans. Automat. Control, 1992, to appear.

[24] A. Haurie, G. Savard, and J. C. White, *A note on an efficient point algorithm for a linear two-stage optimization problem*, Oper. Res., 38 (1990), pp. 553–555.

[25] J. P. Ignizio, *Goal Programming and Extensions*, Lexington Books, Lexington, MA, 1976.

[26] R. G. Jeroslow, *The polynomial hierarchy and a simple model for competitive analysis*, Math. Programming, 32(1985), pp. 146–164.

[27] J. J. Judice and A. M. Faustino, *The solution of the linear bilevel programming problem by using the linear complementarity problem*, Investigação Oper., 8 (1988), pp. 77–95.

[28] ———, *A sequential* LCP *method for bilevel linear programming*, Res. Report, Department of Mathematics, University of Coïmbra, Coïmbra, Portugal, 1989.

[29] L. J. Leblanc and D. E. Boyce, *A bilevel programming algorithm for exact solution of the network design problem with user-optimal flows*, Transportation Res., 20B (1986), pp. 259–265.

[30] P. Marcotte, *Network design problem with congestion effects: A case of bilevel programming*, Math. Programming, 34 (1986), pp. 142–162.

[31] R. E. Marsten, *The design of the* XMP *linear programming library*, Trans. Math. Software,

7 (1981), pp. 481–497.

[32] K. G. MURTY, *Solving the fixed-charge problem by ranking the extreme points*, Oper. Res., 16 (1968), pp. 268–279.

[33] P. M. PARDALOS AND J. B. ROSEN, *Constrained Global Optimization: Algorithms and Applications*, Lecture Notes in Computer Science 268, Springer-Verlag, Berlin, 1987.

[34] G. PAPAVASSILOPOULOS, *Algorithms for static Stakelberg games with linear costs and polyhedral constraints*, in Proc. 21st IEEE Conference on Decisions and Control, 1982, pp. 647–652.

[35] G. SAVARD, *Contributions à la programmation mathématique à deux niveaux*, Ph.D. Thesis, École Polytechnique de Montréal, Montréal, Canada, April 1989.

[36] H. A. TAHA, *Integer Programming, Theory, Applications and Computations*, Academic Press, New York, 1975.

[37] G. ÜNLÜ, *A linear bilevel programming algorithm based on bicriteria programming*, Comput. Oper. Res., 14 (1987), pp. 173–179.

[38] R. E. WENDELL, *Multiple criteria decision making with multiple decision makers*, Oper. Res., 28 (1980), pp. 1100–1111.