Annals of Operations Research 34(1992)89-106

A SEQUENTIAL LCP METHOD FOR BILEVEL LINEAR PROGRAMMING

J.J. JÚDICE
Departamento de Matemática, Universidade de Coimbra, 3000 Coimbra, Portugal

and

A.M. FAUSTINO
Departamento de Engenharia Civil, Universidade do Porto, 4000 Porto, Portugal

Abstract

In this paper, we discuss an SLCP algorithm for the solution of Bilevel Linear Programs (BLP) which consists of solving a sequence of Linear Complementarity Problems (LCP) by using a hybrid enumerative method. This latter algorithm incorporates a number of procedures that reduce substantially the search for a solution of the LCP or for showing that the LCP has no solution. Computational experience with the SLCP algorithm shows that it performs quite well for the solution of small- and medium-scale BLPs with sparse structure. Furthermore, the algorithm is shown to be more efficient than a branch-and-bound method for solving the same problems.

1. Introduction

A Bilevel Linear Program (BLP) can be defined as

$$\begin{aligned} &\underset{x \in \mathbb{R}^m}{\text{minimize}} \quad c^T y + d^T x \\ &\text{subject to} \quad x \geq 0 \\ &\qquad \underset{y \in \mathbb{R}^n}{\text{minimize}} \quad a^T y \\ &\qquad \text{subject to} \quad A_1 y + A_2 x \geq b, \\ &\qquad\qquad x \geq 0, \quad y \geq 0, \end{aligned} \tag{1}$$

where c, $a \in \mathbb{R}^n$, $d \in \mathbb{R}^m$, $A_1 \in \mathbb{R}^{r \times n}$, $A_2 \in \mathbb{R}^{r \times m}$ and $b \in \mathbb{R}^r$.

The BLP has emerged as an important hierarchical optimization problem. A large number of applications has occurred primarily in economic planning. Examples of important applications of the BLP or Nonlinear Bilevel Program are described in [4, 10-12].

A number of algorithms have been developed for finding a global optimum of the BLP. Among relevant procedures, we distinguish a penalty method [2],

branch-and-bound methods [3,7], and an SLCP approach first introduced in [5]. However, the algorithm was not able to solve the BLP in all cases. Júdice and Faustino [9] modified that algorithm to fulfill such a goal. In this paper, we introduce some improvements to the latter convergent SLCP algorithm.

Computational experience with our SLCP algorithm on the solution of small- and medium-scale BLPs ($n \leq 150$, $m \leq 300$) indicates that the modifications stated in this paper have improved the robustness of the algorithm to solve problems of these sizes. The SLCP algorithm is shown to be competitive with the branch-and-bound method for BLPs of small dimensions and becomes much more efficient when the dimension of the BLP increases.

The organization of the paper is as follows. In sections 2 and 3, the SLCP method is described together with the improvements presented in this paper. The Bard and Moore branch-and-bound method is briefly discussed in section 4. Finally, computational experience with both algorithms and the conclusions drawn from our study are presented in the last section.

2. An SLCP algorithm for the BLP

Consider the BLP as stated in (1) and the constraint set

$$H = \{(y, x) \in \mathbb{R}^{n \times m} : A_1 y + A_2 x \geq b, y \geq 0, x \geq 0\}.$$

A global minimum $(\bar{y}, \bar{x})$ of the BLP is also an optimal solution of the following Minimum Linear Complementarity Problem (MLCP) [5]:

$$\begin{aligned} \text{minimize} \quad & c^T y + d^T x \\ \text{subject to} \quad & \alpha = -b + A_1 y + A_2 x, \\ & \beta = a - A_1^T u, \\ & x, y, u, \alpha, \beta \geq 0, \quad y^T \beta = u^T \alpha = 0. \end{aligned} \tag{3}$$

In [9], we proposed a modification of the Bialas and Karwan algorithm [5] for the solution of the MLCP (3). In this method, a parameter λ is introduced and the objective function is replaced by the constraint $c^T y + d^T x \leq \lambda$ to obtain the following parametric LCP:

LCP(λ):

$$\begin{bmatrix} \alpha \\ \beta \\ v_0 \end{bmatrix} = \begin{bmatrix} -b \\ a \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \lambda + \begin{bmatrix} 0 & A_1 & A_2 \\ -A_1^T & 0 & 0 \\ 0 & -c^T & -d^T \end{bmatrix} \begin{bmatrix} u \\ y \\ x \end{bmatrix}, \tag{4}$$

$$x, y, u, \alpha, \beta, v_0 \geq 0, \quad y^T \beta = u^T \alpha = 0.$$

The global minimum $(\bar{y}, \bar{x})$ of the BLP is the solution of the LCP($\bar{\lambda}$), where $\bar{\lambda}$ is the smallest value of λ such that LCP(λ) has a solution. To find $(\bar{y}, \bar{x})$, the method starts by solving the LCP(λ_0) obtained from LCP(λ) by omitting the constraint $c^T y + d^T x \leq \lambda$. This is done by a procedure to be explained in the next section. Let (y^0, x^0) be the solution of this LCP and let $\lambda_0 = c^T y^0 + d^T x^0$. Then the algorithm solves a sequence of LCPs(λ_k), where $\{\lambda_k\}$ is a decreasing sequence defined by

$$\lambda_k = c^T y^{k-1} + d^T x^{k-1} - \gamma | c^T y^{k-1} + d^T x^{k-1} |, \tag{5}$$

with (y^{k-1}, x^{k-1}) being a solution of LCP(λ_{k-1}) and γ a small positive number. The method terminates at iteration k such that LCP(λ_k) has no solution. When this occurs, the solution (y^{k-1}, x^{k-1}) of the LCP(λ_{k-1}) satisfies

$$0 \leq c^T y^{k-1} + d^T x^{k-1} - \text{VAL} \leq \gamma | c^T y^{k-1} + d^T x^{k-1} |, \tag{6}$$

where VAL is the value of the objective function of the BLP at the optimal solution. Hence, if the BLP has a global optimal solution, the SLCP algorithm finds an ε-optimal solution of the BLP, where:

$$\varepsilon = \gamma | c^T y^{k-1} + d^T x^{k-1} |. \tag{7}$$

In practice, if γ is quite small, the solution (y^{k-1}, x^{k-1}) of the last LCP(λ_{k-1}) is in many cases a global minimum of the BLP. In the last section, we discuss the value that γ should take in practice.

The steps of the SLCP algorithm are as follows:

Step 0: Let $k = 0$.

General step: Solve the LCP(λ_k). If LCP(λ_k) has no solution, go to Exit. Otherwise, let (y^k, x^k) be the solution of this LCP. Set

$$\lambda_{k+1} = c^T y^k + d^T x^k - \gamma | c^T y^k + d^T x^k |, \tag{8}$$

where γ is a fixed positive value.

Set $k = k + 1$ and repeat.

Exit: If $k = 0$, then the BLP is infeasible, that is, H is empty. Otherwise (y^{k-1}, x^{k-1}) is an ε-global minimum of the BLP, where ε is given by (7).

The efficiency of the SLCP algorithm depends essentially on the procedure to solve the LCP(λ_k). There exist a number of algorithms for the solution of linear complementarity problems [14]. However, these methods cannot be applied for solving the LCP(λ_k), since this last problem is more general than the traditional linear complementarity problem. In the next section, we describe a hybrid enumerative

method for the solution of LCP(λ_k) that arise in the SLCP algorithm, and we discuss its applicability to the solution of small- and medium-scale BLPs with sparse structure.

3. A hybrid enumerative method for the LCP(λ_k)

In this section, we first describe a hybrid enumerative method proposed in [9] capable of solving the LCP(λ_k) that is required by the SLCP algorithm. Then we discuss three modifications in the enumerative method that makes the SLCP algorithm more robust. Consider the LCP(λ_k) in the form:

$$w = q + Mz + Nx, \tag{9}$$

$$w \geq 0,\ z \geq 0,\ x \geq 0, \tag{10}$$

$$z_i w_i = 0, \qquad i = 1, 2, \ldots, r + n, \tag{11}$$

where $w \in \mathbb{R}^{r+n+1}$, $z \in \mathbb{R}^{r+n}$ and $x \in \mathbb{R}^m$. As in linear programming, a solution (z, w, x) satisfying the linear constraints (9) and (10) is called *feasible*. A solution is *complementary* if the variables z_i and w_i satisfy (11). Variable z_i is said to be the *complement* (or *complementary variable*) of w_i and conversely. An enumerative method attempts to find a complementary solution by using only basic feasible solutions of the system (9). To achieve this, the tree in fig. 1 is explored, where $i_1, i_2, \ldots$ are

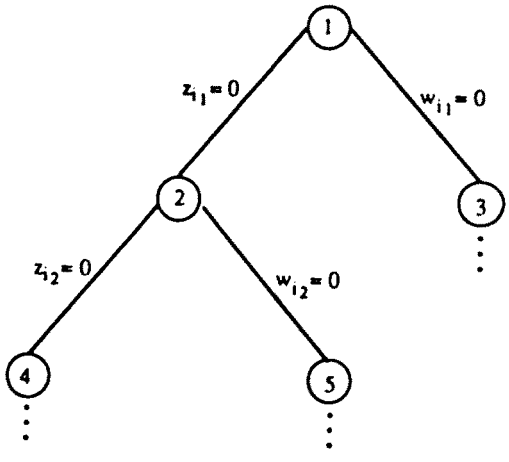

Fig. 1.

integer numbers of $\{1, \ldots, r + n\}$. An initial feasible solution is obtained at node 1 by solving the linear program

$$\begin{aligned} &\text{minimize} && z_0 \\ &\text{subject to} && w = q + pz_0 + Mz + Nx \\ &&& z_0,\ z,\ x,\ w \geq 0, \end{aligned} \tag{12}$$

where z_0 is an artificial variable and p is a nonnegative vector satisfying $p_i > 0$ for all i such that $q_i < 0$. This linear program is solved by a modification of the Phase 1 with a single artificial variable [13]. This modified procedure is described in [9] and consists of minimizing the variable z_0 in such a way that whenever possible the entering variable is chosen among the nonbasic variables whose complement is also nonbasic. By doing this, we control the number of pairs of basic complementary variables, and even a complementary solution can be obtained at the end of this procedure.

Each one of the nodes k with $k \geq 2$ is generated by solving a subproblem that consists of minimizing a variable z_i or w_i subject to the linear constraints of the LCP and some constraints $z_j = 0$ or $w_j = 0$. For instance, to generate node 4 of the tree of fig. 1, it is necessary to solve the linear program

$$\begin{aligned} &\text{minimize} && z_{i_2} \\ &\text{subject to} && w = q + Mz + Nx, \qquad z \geq 0,\ x \geq 0,\ w \geq 0, \\ &&& z_{i_1} = 0. \end{aligned} \tag{13}$$

Such a linear program is solved by a modification of Phase 2 of the simplex method that exploits the same idea of controlling the number of pairs of basic complementary variables. Two cases may occur:

(i) If the variable minimized has value equal to zero, then it is fixed at zero in all descendent paths of the tree.

(ii) If the minimum value of the variable is positive, then the branch is pruned and the node is fathomed.

The enumerative method attempts to solve the LCP by generating successive nodes of the tree, according to the process explained above. The algorithm either finds a solution for the LCP (it is the first complementary feasible solution) or establishes that the LCP has no solution (all the generated nodes of the tree are fathomed).

The enumerative method is efficient only if few nodes are generated before a complementary solution is found or it is verified that none exists. There are some heuristic rules and some procedures that usually improve the efficiency of the algorithm. The heuristic rules relate to the choice of the node of the tree and of the branching pair (z_{i_k}, w_{i_k}). These are detailed in [9]. The algorithm for generating nodes described before is an example of a procedure that reduces the search for a

complementary solution. Next, we briefly discuss another technique that was introduced in [1] and is applied in each node for the same purpose of reducing the overall search.

Al-Khayyal's algorithm [1] is a modified reduced gradient method that finds a local star minimum of the function

$$f(z, w, x) = \sum_{i=1}^{r+n} z_i w_i, \tag{14}$$

that is, a basic feasible solution $(\bar{z}, \bar{w}, \bar{x})$ such that

$$f(\bar{z}, \bar{w}, \bar{x}) \leq f(z, w, x)$$

for all its adjacent basic feasible solutions (z, w, x) of the feasible set of the LCP

$$\{(z, w, x) : w = q + Mz + Nx, z \geq 0, w \geq 0, x \geq 0\}. \tag{15}$$

If $(\bar{z}, \bar{w}, \bar{x})$ is a basic feasible solution and $(\tilde{z}, \tilde{w}, \tilde{x})$ is an adjacent basic feasible solution, then we can write

$$(\tilde{z}, \tilde{w}, \tilde{x}) = (\bar{z}, \bar{w}, \bar{x}) + \mu_0(d^z, d^w, d^x),$$

where μ_0 is the minimum ratio of the simplex method and $d = (d^z, d^w, d^x)$ is a feasible direction such that d^z, d^w and d^x are the vectors of the components of d corresponding to the z, w and x variables, respectively. Due to the special structure of the $\mathbf{LCP}(\lambda_k)$ presented in (4), then [9]

$$f(\tilde{z}, \tilde{w}, \tilde{x}) - f(\bar{z}, \bar{w}, \bar{x}) = \mu_0 \sum_{i=1}^{r+n} (\bar{z}_i d_i^w + \bar{w}_i d_i^z) = -\mu_0 \bar{e}_s,$$

where $\bar{e}_s$ is the reduced-cost of the nonbasic variable of index s associated with the linear function

$$\sum_{i=1}^{r+n} (\bar{z}_i w_i + \bar{w}_i z_i). \tag{16}$$

Therefore, Al-Khayyal's modified reduced-gradient algorithm is in this case a simplex-type method in which the reduced costs $\bar{e}_j$ of the nonbasic variables are associated with the linear function (16). We also incorporate Bland's rule [6] to avoid the occurrence of cycling in the degenerate pivot steps [8].

The reader is referred to [9] for a detailed description of the steps of the hybrid enumerative method that incorporates this last algorithm, the procedures for generation of nodes, and the heuristic rules for choice of nodes and the complementary pairs of variables. The hybrid enumerative method can be implemented for solving

large and sparse LCPs. The implementation uses reinversion and updating techniques for the LU decompositions of the basis matrices [15] used by the simplex-type procedures and special data structures. We recommend [8] for a description of this implementation.

As discussed in section 2, the hybrid enumerative method is used to solve the LCPs(λ_k) required by the SLCP algorithm. For any two values $\lambda_k < \lambda_{k-1}$, the LCP($\lambda_{k-1}$) and LCP($\lambda_k$) differ only in the last component of the vector q. Furthermore, if B is the basis associated with the solution $(\bar{z}, \bar{w}, \bar{x})$ of the LCP(λ_{k-1}) obtained by the hybrid enumerative method and q is the right-hand side of the LCP(λ_k), then the vector $\bar{q} = B^{-1}q$ satisfies

$$\bar{q}_j \geq 0 \quad \text{for all} \quad j = 1, \ldots, r+n \quad \text{and} \quad \bar{q}_{r+n+1} < 0. \tag{17}$$

Hence, the solution $(\bar{z}, \bar{w}, \bar{x})$ of the LCP(λ_{k-1}) can be used as the initial basic solution for the LCP(λ_k). Since the vector $B^{-1}q$ satisfies (17), the vector introduced in (12) can be defined by

$$p = B \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

where $0 \in \mathbb{R}^{r+n}$ is the null vector. The choice of this initial basic solution usually provides great reductions in the computational effort of the hybrid enumerative method. In fact, the algorithm quite often requires a small number of pivot operations to find a solution for the LCP(λ_k).

Below, we introduce three modifications to the hybrid enumerative method described above.

(i) In the procedure for finding a local star minimum of the function (14), we use the same idea of controlling the pair of complementary basic variables previously discussed. Hence, whenever possible, we choose the entering variable from among the set of nonbasic variables such that $\bar{e}_s > 0$ and whose complements are also nonbasic.

(ii) Whenever a solution of the LCP(λ_k) is found, we try to find another solution of the LCP(λ_k) such that $c^T y + d^T x$ has a smaller value. To do this, we try to maximize the variable v_0 in such a way that the complementarity condition $z^T w = 0$ holds in each iteration. This is done by maximizing v_0 under the Basic Restricted Simplex Rule described in [5], that is, we apply the simplex method to maximize v_0, but in each iteration the complement of the entering variable must be nonbasic. This kind of procedure, called MAXVAR, can terminate in two possible ways:

(a) The maximum value of the variable v_0 subject to the linear constraints of the LCP(λ_k) has been achieved. In this case, a global minimum for the BLP has been found and the SLCP algorithm terminates (TERM = 1).

(b) The variable v_0 can still be increased, but only by destroying the complementarity condition, that is, all complements of the nonbasic variables with negative reduced costs are basic. In this case, λ is updated by (8) and another LCP has to be solved in the SLCP algorithm (TERM = 2).

(iii) As discussed in [9], in the last stages of the SLCP algorithm the LCPs tend to be much more difficult to solve. The incorporation of the two procedures stated above usually reduces the computational effort for the solution of the last LCPs required by the SLCP algorithm. Despite this, the difficulty still persists, particularly for BLPs of larger dimensions. Our computational experience with the SLCP algorithm has suggested that in many cases when an LCP is quite difficult, if we reduce slightly the value of λ, then the resulting LCP is much easier. Hence, we have decided to limit the number of pivot steps that the solution of **LCP**(λ) may require by a quantity NMAXPV given by the user. If this number is achieved, then λ_k is updated by

$$\lambda_k = \lambda_k - \bar{\gamma} \mid \lambda_{k-1} \mid, \tag{18}$$

where $\bar{\gamma}$ is a positive constant greater than γ and the **LCP**(λ_k) is solved by starting with the current basic solution. Furthermore, no limit on the number of pivot steps is assumed for this latter LCP. If this LCP has no solution, then the last complementary solution obtained by the SLCP algorithm is an $\bar{\varepsilon}$-optimal solution of the BLP, where

$$\bar{\varepsilon} = \bar{\gamma} \mid c^{\mathrm{T}} y^{k-1} + d^{\mathrm{T}} x^{k-1} \mid. \tag{19}$$

In the last section, we discuss the value that $\bar{\gamma}$ should take in practice.

These two latter modifications lead to the following algorithm, which is used in each iteration k of the SLCP method:

Step 0: Set NMAX = NMAXPV, where NMAXPV is a positive number chosen by the user.

Step 1: Apply the hybrid enumerative method to solve the **LCP**(λ_k). If the number of simplex pivot steps required by the hybrid enumerative method attains NMAX, update λ_k by (18), set NMAX = $+\infty$ (in practice, NMAX is set equal to a very large positive number) and repeat step 1. Otherwise, the hybrid enumerative method terminates and we go to step 2.

Step 2: If **LCP**(λ_k) has no solution, set TERM = 3 and go to Exit. Otherwise, apply the MAXVAR algorithm and go to Exit.

Exit: If TERM = 1, the solution (obtained by the MAXVAR algorithm) is a global minimum for the BLP.

If TERM = 2, the iteration $(k+1)$ of the SLCP algorithm has to be performed with λ_{k+1} given by (8).

If TERM = 3, the solution obtained in the iteration $(k-1)$ of the SLCP algorithm is an ε or an $\bar{\varepsilon}$ global minimum for the BLP, depending on NMAX to be less or equal to $+\infty$, repectively.

The incorporation of these modifications in the SLCP algorithm has produced a more robust algorithm. This is reported in the last section of this paper.

4. A branch-and-bound method for the BLP

As stated in section 1, to gain a better idea of the ability of the SLCP algorithm to solve BLPs, we have decided to compare it with a branch-and-bound method developed by Bard and Moore [3]. In this section, we briefly describe this latter algorithm. We write the MLCP (3) in the form

$$\begin{aligned} &\text{minimize} \quad f^T z + d^T x \\ &\text{subject to} \quad w = q + Mz + Nx, \\ &\qquad z \geq 0,\ w \geq 0,\ x \geq 0,\ z^T w = 0, \end{aligned} \tag{20}$$

where

$$w = \begin{bmatrix} \alpha \\ \beta \end{bmatrix},\ z = \begin{bmatrix} u \\ y \end{bmatrix},\ M = \begin{bmatrix} 0 & A_1 \\ -A_1^T & 0 \end{bmatrix},\ N = \begin{bmatrix} A_2 \\ 0 \end{bmatrix},\ q = \begin{bmatrix} -b \\ a \end{bmatrix},\ f = \begin{bmatrix} 0 \\ c \end{bmatrix}.$$

The branch-and-bound method starts by solving the linear program obtained from (20) by omitting the complementarity condition $z^T w = 0$. If the optimal solution of this linear program satisfies this complementarity condition, then an optimal solution of the BLP is at hand. Otherwise, there must exist a pair of complementary basic positive variables z_i and w_i. The algorithm chooses a pair of such variables (z_{i_1}, w_{i_1}) and generates the nodes of the tree of fig. 1 by minimizing z_{i_1}, w_{i_1} subject to the linear constraints of the MLCP, respectively. In practice, it is advisable to choose the pair for which $z_{i_1} w_{i_1}$ is maximum.

In any iteration of the branch-and-bound method, a node of the tree is picked (in practice, it is advisable to choose the node for which the objective function $f^T z + d^T x$ has the lowest value). After choosing the node, we solve a linear program obtained from (20) by relaxing the complementarity condition and adding some constraints of the form $z_{i_k} = 0$ or $w_{i_k} = 0$ corresponding to the variables that have been fixed in the nodes belonging to a path from the current node up to the root of the tree. As before, this linear program is solved by Phase 2 of the simplex method starting from the basic solution corresponding to this particular node. If the optimal solution of this linear program satisfies the complementarity condition, then it is stored as the best solution and is called an incumbent. All the nodes for which the value of $f^T z + d^T x$ is larger than or equal to the value of the objective function corresponding to the incumbent are discarded (they are fathomed). If no unfathomed

nodes exist, the algorithm terminates with a global solution given by the last incumbent. Otherwise, the procedure is repeated.

From this brief description, it is not difficult to see that the implementation for the hybrid enumerative method can be used with minor modifications for the branch-and-bound method. We developed such an implemenation for solving small- and medium-scale sparse BLPs. As before, the linear programs are solved by an implementation of the simplex method for large-scale linear programs, which is described in [15].

In theory, the branch-and-bound method finds a global minimum of the BLP are there is no reason to assume that this method is more or less efficient than the SLCP algorithm. Both methods require an implicit search for a global minimum, whence the computational effort grows exponentially with the dimension of the BLP. However, in the branch-and-bound method there is a conflict between the solution of the linear programs required in each node and the generation of the nodes. In fact, when we solve a linear program in a node the value of $z^T w$ can be increased, while the value of the objective function $f^T z + d^T x$ can be increased when a node is generated. This conflict is not present in the SLCP algorithm, since only LCPs are solved in each iteration of the algorithm. Furthermore, the SLCP algorithm contains a number of efficient techniques that reduce substantially the search for a global optimal solution of the BLP. So, it seems that the SLCP algorithm will be more efficient than the branch-and-bound method. Computational experience presented in the next section confirms exactly this statement, and the gap increases dramatically with an increase of the dimension of the BLP.

5. Computational experience

In this section, we present some computational experience with the SLCP and branch-and-bound algorithms discussed in this paper on the solution of some small- and medium-scale sparse BLPs. Two sets of test problems have been considered. The first set contains problems with the same characteristics of those described in [7], while the second set has been described in [9]. As discussed in that paper, we have associated with each matrix $A = [A_1, A_2]$ two test problems simulating conflicting (problems TPG) and nonconflicting (problems TPN) situations.

The characteristics of the test problems and the experimental results for the SLCP and branch-and-bound algorithms are presented in tables 1, 2, 3 and 4. All the tests have been performed on a CDC CYBER 180-830 at the University of Porto. The parameters included in the tables have the following meanings:

n = number of second-level variables y,

m = number of first-level variables x,

r = number of constraints = number of rows of $A = [A_1, A_2]$,

rsp = relative sparsity of $A = [A_1, A_2]$ = (number of nonzeros of A)/$(n + m)$,

psp = sparsity percentage of A = [(number of nonzeros of A)/($r \times (n + m)$)] × 100,
nrow = number of rows of the MLCP = $r + n$,
ncol = number of columns of the MLCP = $r + n + m$,
NLCP = number of LCPs(λ_k) to be solved by the SLCP algorithm,
ND = number of nodes required by the branch-and-bound method,
NI = total number of simplex pivot operations required by the SLCP or branch-and-bound algorithms,
NS = the algorithm has not been able to terminate after 20,000 simplex pivot steps,
T = CPU time in seconds for the SLCP and branch-and-bound algorithms.

The first aim of our computational study was to investigate the importance of the modifications stated in this paper on the efficiency of the SLCP algorithm. To do this, the test problems described in [9] were solved by the modified SLCP method with NMAXPV = 500, $\gamma = 0.01$ and $\bar{\gamma} = 0.05$. We denote by SLCP METHOD 1 such an algorithm to distinguish it from the SLCP METHOD 2 in which $\gamma = 0.001$ and $\bar{\gamma} = 0.01$.

A comparison between the results of the SLCP METHOD 1 presented in table 1 and those of the table of [9] leads to the conclusion that the modifications described in this paper improve the efficiency of the SLCP algorithm.

The SLCP algorithm is designed to find an ε-optimal solution of the BLP. If the algorithm terminates with the MAXVAR procedure, then a global minimum is achieved. Otherwise, only ε-optimality can be assured in theory. Hence, it is interesting to investigate the influence that a reduction of the value of ε (γ and $\bar{\gamma}$) has on the efficiency of the SLCP method. The SLCP METHOD 2 was considered for such a purpose. In table 1, we have added a column headed by OPT in which the value α means that the SLCP method can assure, in theory, at least α% of the optimum ($\alpha = 0$ means that a global minimum is at hand). In the case of $\alpha \neq 0$, it seems important to verify whether the solution obtained by the SLCP method is a global minimum. To see this, we have solved the same problems by the branch-and-bound method. We have written Y (N) when the solutions of the two methods agree (do not agree), that is, when the SLCP algorithm has (has not) found a global optimum. The branch-and-bound method could not solve the BLPs of larger dimensions in reasonable time (less than 20,000 simplex pivot operations). Hence, we cannot see whether the solution found by the SLCP method is the global minimum for these BLPs. We write a question mark in the column *OPT* for these test problems.

The results presented in table 1 lead to the following conclusions:

(i) Both versions of the SLCP method find an ε-optimal solution for all the test problems. The algorithms are quite fast in obtaining an ε-optimal solution (see the columns headed by OPTIMAL). However, the algorithms face some difficulties to terminate for BLPs of larger dimensions.

Table 1

Júdice and Faustino [9] test problems

	Dimensions						SLCP METHOD 1						SLCP METHOD 2					
							Total			Optimal			Total			Optimal		
Problem	*n*	*m*	*r*	*rsp*	*nrow*	*ncol*	*NLCP*	*NI*	*T*	*NI*	*T*	*OPT*	*NLCP*	*NI*	*T*	*NI*	*T*	*OPT*
TPN1	30	50	30	3.5 – 5.0	60	110	5	35	0.8	35	0.8	0, Y	5	37	0.9	37	0.9	0, Y
TPG1							7	130	3.4	130	3.4	0, Y	8	136	3.7	136	3.7	0, Y
TPN2							7	259	6.8	219	5.5	1, Y	9	400	11.5	335	9.6	0.1, Y
TPG2							10	264	8.5	264	8.5	0, Y	12	293	10.8	293	10.8	0, Y
TPN3	50	120	50	5.2 – 7.5	100	220	7	132	6.1	132	6.1	0, Y	10	280	14.1	280	14.1	0, Y
TPG3							6	304	18.7	304	18.7	0, Y	9	348	23.9	348	23.9	0, Y
TPN4							15	729	45.3	400	25.3	1, N	17	701	47.2	377	25.8	0.1, Y
TPG4							26	1215	90.6	1127	85.0	1, Y	26	1068	86.1	730	62.4	1, Y
TPN5	100	300	100	5.3 – 7.1	200	500	11	655	66.2	655	66.2	0, Y	23	1115	119	1112	119	0.1, N
TPG5							14	1031	141	1023	140	1, N	21	1775	284	1775	284	0, Y
TPN6							4	585	86.8	585	86.8	0, Y	5	595	88.7	595	88.7	0, Y
TPG6							8	1744	396	1744	396	0, Y	5	1298	287	1298	287	0, Y
TPN7	150	250	150	5.3 – 7.1	300	550	19	2183	278	1364	181	5, Y	31	6657	856	4763	618	1, Y
TPG7							24	2653	470	2011	395	5, ?	39	4257	776	2498	551	1, ?
TPN8							17	2332	352	1757	275	5, N	40	6910	1040	2380	382	1, ?
TPG8							30	6378	1571	5285	1383	5, ?	69	8843	2123	7879	1980	5, ?

(ii) The SLCP METHOD 1 performs better than the SLCP METHOD 2. This is not surprising, since the latter version uses smaller values for γ and $\bar{\gamma}$. However, the difference is not very large. Furthermore, the SLCP METHOD 2 can assure, in theory, a solution that is closer to the global optimum.

(iii) For all the test problems but one, the SLCP METHOD 2 was able to find a global minimum (even when $\alpha > 0.1$). The SLCP METHOD 1 has failed in three cases. There is a test problem in which the SLCP METHOD 1 has found the global minimum, but the second version failed.

(iv) The value $\bar{\gamma} = 0.05$ had to be used for the SLCP METHOD 2 to terminate problem TPG8 in less than 20,000 simplex pivot operations.

These conclusions lead to our recommendation to use the version of the SLCP method in which $\gamma = 0.001$ and $\bar{\gamma} = 0.01$. However, when $(r + n)$ is large, it is more advisable to set $\bar{\gamma} = 0.05$.

The value NMAXPV = 500 was used in the experimentations whose results are presented in table 1. To gain a better idea of the value that NMAXPV should take, we have decided to allow an increase of this quantity in the SLCP METHOD 2 for all the test problems in which the algorithm could assure only a percentage of the optimum superior to 0.1. We have run these test problems with NMAXPV equal to 1500, 3000, 4500.

The results presented in table 2 show that in terms of speed, too large values for NMAXPV are not appropriate. However, NMAXPV = 500 is not the best choice for a problem in which the number $(r + n)$ of complementary variables is equal to 300. So, if we are only interested in the speed of the procedure, then a relatively small value of NMAXPV ($\leq 10\ (r + n)$) should be used and $\bar{\gamma}$ should be set equal to 0.05 for large values of $(r + n)$. If only precision is at stake, then NMAXPV can take larger values and $\bar{\gamma} = 0.01$ is a good choice. Based on these and other computational results, we recommend the SLCP method incorporating the modifications stated in this paper with the following values:

$$\text{NMAXPV} = 10\ (r + n), \quad \gamma = 0.001, \quad \bar{\gamma} = 0.01. \tag{21}$$

and $\bar{\gamma} = 0.05$ when $(r + n)$ is large.

In tables 3 and 4, the efficiencies of the SLCP algorithm with the values given by (21) and the branch-and-bound method are compared. We also test another version of the branch-and-bound method in which the initial solution (and incumbent) is given by the SLCP algorithm (it is denoted by Incumbent B.B). To perform the comparative study, we have generated test problems in the following way:

(i) Each test problem TP*i* has associated certain values of r, n, m and degree of sparsity.

(ii) For each TP*i*, five test problems are considered in which the matrices are generated by a technique similar to that in [9] (table 3) and in [7] (table 4).

Table 2

Influence of NMAXPV on the SLCP algorithm

Problem	$r+n$	NMAXPV	Total			Optimal		*OPT*
			NLCP	*NI*	*T*	*NI*	*T*	
TPG4	100	500	26	1068	86.1	730	62.4	1, Y
		1500	25	2212	163	730	62.4	0.1, Y
		3000	25	2212	163	730	62.4	0.1, Y
		4500	25	2212	163	730	62.4	0.1, Y
TPN7	300	500	31	6657	856	4763	618	1, y
		1500	29	5404	691	2541	332	1, Y
		3000	28	4979	638	2541	332	0.1, Y
		4500	28	4979	638	2541	332	0.1, Y
TPG7	300	500	39	4257	776	2498	551	1, ?
		1500	39	5084	871	2498	551	1, ?
		3000	39	6574	1051	2498	551	1, ?
		4500	38	7018	1090	2498	551	0.1, ?
TPN8	300	500	40	6910	1040	2380	382	1, N
		1500	43	10,621	1568	5166	777	1, ?
		3000	43	12,161	1789	5166	777	1, ?
		4500	43	9741	1430	5166	777	1, ?
TPG8	300	500	69	8843	2123	7879	1980	5, ?
		1500	69	9854	2340	7879	1980	5, ?
		3000	69	11,542	2660	7879	1980	5, ?
		4500	69	13,016	2952	7879	1980	5, ?

Tables 3 and 4 contain the information of the behavior of the algorithms for solving these five test problems of each TP*i* by showing the best (B), worst (W) and average (A) performance in terms of simplex pivot operations. As before, the information concerning the SLCP method contains a column headed by *OPT* with similar meanings for the rows B and W. The numbers that appear in the column *OPT* and rows A represent the number of times in which the SLCP method has found the global minimum. If for a TP*i* there is at least a test problem for which an algorithm has not been able to terminate in less than 20,000 simplex pivot operations, then we write in row A the number of times that the algorithm has successfully terminated.

The results presented in tables 3 and 4 lead to the following conclusions:

(i) The SLCP algorithm is quite efficient to find an ε-optimal solution (see column Optimal). However, it faces difficulties to terminate for BLPs of larger dimensions.

Table 3

Solution of BLPs generated as in [9]

Prob.	Dimensions								SLCP						Branch-and-bound			Incumbent B.B			
									Total			Optimal									
	n	*m*	*r*	*rsp*	*nrow*	*ncol*	NMAXPV		*NLCP*	*NI*	*T*	*NI*	*T*	*OPT*	*ND*	*NI*	*T*	*NLCP*	*ND*	*NI*	*T*
TPN2	30	50	30	5.0	60	110	600	B	4	75	2.0	75	2.0	0, Y	9	108	2.7	4	0	75	2.0
								A	8	299	8.7	262	8.3	3	41	427	12.8	8	5	318	9.4
								W	10	774	23.3	774	23.3	0, Y	99	1239	38.7	10	0	774	23.3
TPG2	30	50	30	5.0	60	110	600	B	13	205	6.7	203	6.6	0.1, Y	13	138	4.2	9	7	238	7.8
								A	15	406	13.3	405	13.3	4	49	373	11.4	15	4	420	13.8
								W	29	1032	32.3	1032	32.3	0, Y	173	1023	31.6	29	0	1032	32.3
TPN4	50	120	50	7.5	100	220	1000	B	3	99	5.5	99	5.5	0, Y	3	93	4.8	3	0	99	5.5
								A	11	1719	111	662	43.2	5	135	4309	290	11	107	3119	201
								W	17	4116	270	1428	93.9	1, Y	399	15,284	1045	17	295	8323	551
TPG4	50	120	50	7.5	100	220	1000	B	9	435	39.1	435	39.1	0, Y	11	478	36.8	9	0	435	39.1
								A	18	2681	190	651	54.3	5	159	3744	261	18	126	4411	302
								W	17	8275	565	735	57.1	1, Y	407	10,245	738	17	351	13,232	891
TPN6	100	300	100	7.1	200	500	2000	B	5	517	67.0	517	67.9	0, Y	7	355	44.7	5	0	517	67.0
								A	21	5828	746	3318	431	4		3			4		
								W	36	14,799	1873	6737	872	1, ?		NS			NS		
TPG6	100	300	100	7.1	200	500	2000	B	5	1298	287	1298	287	0, Y	5	789	112	5	0	1298	287
								A	23	6664	943	3907	604	4		4			4		
								W	32	25,590	3266	11,970	1588	1, ?		NS			NS		
TPN8	150	250	150	7.1	300	550	3000	B	43	8289	1240	5166	783	5, ?		NS			NS		
								A			3					0			0		
								W		NS		4015	702	–		NS			NS		
TPG8	150	250	150	7.1	300	550	3000	B	40	5444	1500	3969	1248	0.1, N	129	9171	1803	40	131	11,179	2428
								A	44	10,140	2282	7454	1752	–		1			1		
								W	41	16,779	3410	13,481	2828	5, ?		NS			NS		

Table 4

Solution of BLPs generated as in [7]

	Dimensions								SLCP						Branch-and-bound			Incumbent B.B			
									Total			Optimal									
TP	*n*	*m*	*r*	*psp*	*nrow*	*ncol*	NMAXPV		*NLCP*	*NI*	*T*	*NI*	*T*	*OPT*	*ND*	*NI*	*T*	*NLCP*	*ND*	*NI*	*T*
								B	3	94	3.5	94	3.5	0, Y	5	68	2.2	3	0	94	3.5
1	20	100	28	17	48	148	480	A	6	161	6.3	160	6.2	3	13	146	4.9	6	3	225	6.6
								W	7	225	9.0	223	9.0	0.1, Y	17	210	7.4	6	3	233	9.4
								B	5	173	10.2	173	10.2	0, Y	9	153	7.9	5	0	173	10.2
2	20	100	28	40	48	148	480	A	9	252	14.8	249	14.7	3	12	263	13.9	9	5	288	16.8
								W	13	312	20.0	305	19.7	0.1, N	23	491	25.8	12	21	465	28.4
								B	19	1237	59.7	956	46.8	0.1, N	65	1149	53.1	19	45	1772	83.4
3	40	60	40	17	80	140	800	A	22	2337	117	685	36.8	3	384	7425	329	22	359	6276	307
								W	21	2793	133	623	34.7	1, Y	825	11,829	541	27	541	9866	480
								B	22	864	82.0	626	58.1	0.1, Y	55	1569	133	22	67	1672	150
4	40	60	40	40	80	140	800	A	24	2709	206	801	71.3	4		4		24	381	7403	528
								W	18	6637	458	641	55.1	1, Y		NS		18	1115	18,470	1220
								B	18	935	55.1	777	47.0	0.1, N	115	2081	109	18	51	1362	78.1
5	40	120	40	17	80	200	800	A	18	2096	125	1171	72.0	4	486	7089	386	18	282	4878	287
								W	16	3435	204	2226	134	1, Y	925	15,900	871	16	499	8134	475
								B	14	507	59.0	507	59.0	0, Y	5	264	23.5	14	0	507	59.0
6	40	120	40	40	80	200	800	A	17	1106	114	600	68.2	3	114	2502	215	17	86	2118	202
								W	22	1665	153	637	69.0	1, Y	251	5350	487	15	213	4526	438
								B	38	3154	417	1634	259	5, ?		NS				NS	
7	70	80	70	17	140	220	1400	A	35	3624	477	1901	284	–		0				0	
								W	29	4196	623	2514	355	5, ?		NS				NS	
								B	18	923	259	907	255	0.1, N	11	1041	240	18	3	985	270
8	70	80	70	40	140	220	1400	A	38	3781	930	2380	585	–		1				1	
								W	48	4851	1110	3186	741	5, ?		NS				NS	

(ii) The branch-and-bound method is competitive with the SLCP method for BLPs of small dimensions (in some cases, it is even more efficient), but it is significantly less efficient for BLPs of larger dimensions.

(iii) The use of an incumbent given by the SLCP method improves the efficiency of the branch-and-bound method for BLPs of larger dimensions. However, even in this case the branch-and-bound method is not competitive with the SLCP algorithm.

(iv) The SLCP algorithm has found the global minimum of the BLP in more than 60% of the test problems.

The computational study presented in this paper shows that the branch-and-bound method as stated in [3] is not an efficient procedure to solve BLPs of reasonable dimensions. Furthermore, the use of an incumbent given by the SLCP algorithm improves the efficiency of the branch-and-bound method. In practice, a criterion is required that allows the user to move from the SLCP algorithm to the branch-and-bound method. Since our experience has shown that the hybrid enumerative method only requires a large number of simplex pivot operations in the last stages of the SLCP algorithm, then such a criterion can be designed by using the quantity NMAXPV introduced in section 3. So, we recommend the user to fix initially NMAXPV and to apply the SLCP algorithm. If the number of simplex pivot operations required by the hybrid enumerative method in an iteration $k \geq 1$ of the SLCP algorithm attains NMAXPV, then the solution obtained in the iteration $(k-1)$ is an incumbent for the branch-and-bound method and this method starts from this basic solution. Such a procedure for finding a good incumbent for the branch-and-bound method can be called a truncated SLCP algorithm. The experience shows that the value of NMAXPV should not be chosen either too large or too small. NMAXPV = $10\,(r+n)$ is probably a good choice.

Computational experience with larger BLPs has shown that the truncated SLCP algorithm is not sufficient to provide a robust branch-and-bound method for the BLP. In our opinion, the method lacks good lower bounds. The generation of good lower bounds for the branch-and-bound method is an important research aspect and deserves some attention.

Recently, Hansen et al. [7] have developed a new branch-and-bound method for the BLP. The algorithm is based on the idea of eliminating the variables y in the follower's problem in order to reduce the complexity of the BLP. This is done at the expense of the generation of many 0–1 logical variables. A branch-and-bound method is then applied to solve the resulting 0–1 mixed integer programming problem. We have not tested the algorithm, but it seems that in general the method is more efficient than the Bard and Moore [3] branch-and-bound algorithm, particularly for sparse BLPs [7]. However, the computational experience in that paper indicates that the method is not very robust. We believe that the robustness of the method can be improved if an upper bound is first found by the truncated SLCP algorithm.

References

[1] F.A. Al-Khayyal, An implicit enumeration procedure for the general linear complementarity problem, Math. Progr. Studies 31(1987)1-20.

[2] G. Anandalingam and D.J. White, A penalty function approach for solving bilevel linear programs, Department of Systems, University of Pennsylvania (1988).

[3] J.F. Bard and J.J. Moore, A branch-and-bound algorithm for the bilevel linear program, SIAM J. Sci. Statist. Comput. 11(1990)281-292.

[4] O. Ben-Ayed, C.E. Blair and D.E. Boyce, Solving a real world highway network design problem using bilevel linear programming, BEBR Faculty Working Paper 1463, University of Illinois at Urbana-Champaign (1988).

[5] W.F. Bialas and M.H. Karwan, Two-level linear programming, Manag. Sci. 30(1984)1004-1020.

[6] R.G. Bland, New finite pivoting rules for the simplex method, Math. Oper. Res. 2(1977)103-107.

[7] P. Hansen, B. Jaumard and G. Savard, A variable elimination algorithm for bilevel linear programming, RUTCOR, Rutgers University (1989).

[8] J.J. Júdice and A.M. Faustino, An experimental investigation of enumerative methods for the linear complementarity problem, Comput. Oper. Res. 15(1988)417-426.

[9] J.J. Júdice and A.M. Faustino, The solution of the linear bilevel linear programming problem by using the linear complementarity problem, Investigação Operacional 8(1988)77-95.

[10] C.D. Kolstad and L.S. Lasdon, Derivate evaluation and computational experience with large bilevel mathematical programs, Faculty Working Paper 1266, College of Commerce and Business Administration, University of Illinois at Urbana-Champaign (1986).

[11] C.D. Kolstad, A review of the literature on bilevel mathematical programming, Technical Report LA-10284-MS, Los Alamos National Laboratory (1985).

[12] P. Marcotte, Network design problem with congestion effects: A case of bilevel programming, Math. Progr. 34(1986)142-162.

[13] K.G. Murty, *Linear Programming* (Wiley, New York, 1983).

[14] K.G. Murty, *Linear Complementarity, Linear and Nonlinear Programming* (Heldermann, Berlin, 1988).

[15] J.K. Reid, A sparsity-exploitation variant of the Bartels-Golub decomposition of linear programming bases, Math. Progr. 24(1982)55-69.