Annals of Operations Research 34(1992)149-162

AN ALGORITHM FOR THE MIXED-INTEGER NONLINEAR BILEVEL PROGRAMMING PROBLEM*

Thomas A. EDMUNDS
Systems Research Group, Lawrence Livermore National Laboratory, University of California, Livermore, CA 94550, USA

and

Jonathan F. BARD
Department of Mechanical Engineering, Operations Research Group, University of Texas, Austin, TX 78712, USA

Abstract

The bilevel programming problem (BLPP) is a two-person nonzero sum game in which play is sequential and cooperation is not permitted. In this paper, we examine a class of BLPPs where the leader controls a set of continuous and discrete variables and tries to minimize a convex nonlinear objective function. The follower's objective function is a convex quadratic in a continuous decision space. All constraints are assumed to be linear. A branch and bound algorithm is developed that finds global optima. The main purpose of this paper is to identify efficient branching rules, and to determine the computational burden of the numeric procedures. Extensive test results are reported. We close by showing that it is not readily possible to extend the algorithm to the more general case involving integer follower variables.

1. Introduction

The bilevel programming problem (BLPP) can be viewed as a static Stackelberg game in which control of the decision variables is partitioned between two players who seek to minimize their individual objective functions [7, 18]. Play is sequential and noncooperative. Perfect information is assumed in that either player knows the objective function and allowable strategies of the other.

This type of leader-follower game can be used to model many hierarchical systems in which two autonomous agents make decisions in a prescribed manner. Applications can be found in such areas as government regulation [10] and decentralized control [3]. In addition, decomposition procedures can be interpreted to fit the format of these games [11].

*This work was supported by a grant from the Advanced Research Program of the Texas Higher Education Coordinating Board.

Throughout this paper, we examine BLPPs with convex quadratic objective functions, linear constraints, and both continuous and binary decision variables. In the next section, the BLPP is presented together with its single-level equivalent formulation. Here, the leader is given control of all binary decision variables. A branch and bound algorithm for solving the resultant problem is given in section 3. Various branching techniques are examined in section 4 and computational results are presented. In section 5, we show that a branch and bound approach may fail if binary variables are introduced into the follower's problem. Finally, we conclude with an assessment of the methodology.

2. Single-level equivalent

In the model, it is assumed that the leader moves first and selects both x, an n_1-dimensional vector of continuous variables, and w, an n_2-dimensional vector of binary variables, in an attempt to minimize a nonlinear function $F(x, w, y(x, w))$. This notation stresses the fact that the leader's problem is implicit in the follower's variables y. Having observed the leader's choice, the follower reacts by selecting y, and n_3-dimensional vector of continuous variables, to minimize his nonlinear objective function $f(x, w, y)$. Note that the leader's choice of strategy affects both the follower's objective and allowable decisions, and that the follower's choice affects the leader's objective.

The BLPP corresponding to this game takes the following form:

$$\underset{x,\,w}{\text{minimize}} \quad F(x, w, y(x, w)), \qquad \text{where } y \text{ solves} \tag{1a}$$

$$\underset{y}{\text{minimize}} \quad f(x, w, y) \tag{1b}$$

$$\text{subject to} \quad g(x, w, y) \leq 0, \tag{1c}$$

$$w_l \in \{0, 1\}, \qquad l = 1, \ldots, n_2, \tag{1d}$$

where $g(x, w, y)$ is an m-dimensional vector-valued function.

If F, f and g are linear and $n_2 = 0$ (no discrete variables), the resulting formulation of (1) has been solved using several approaches. One approach involves vertex search [8,9], while a second approach begins by replacing the follower's problem (1b) and (1c) with his Kuhn–Tucker conditions to form a standard optimization problem, and then using some type of implicit enumeration to find the solution [5,6,13]. In addition, a penalty function approach has been examined by Anandalingam and White [2] and a variable elimination algorithm was developed by Hansen et al. [14]. Finally, Júdice and Faustino [15] reformulated the BLPP as a linear complementary problem to obtain a solution.

A few versions of (1) with continuous variables and nonlinear F, f and g have been solved. Aiyoshi and Shimuzu [1] developed a penalty method for such problems but were able to handle only small test cases. Their approach was hampered by slow

convergence and an inability to verify global optimality regardless of model structure. More recently, Bard [4] proposed an efficient branch and bound scheme for the convex case where F and f are quadratic and g is linear; Edmunds [11] extended this work to deal with more general forms. Others, including Basar and Selbuz [7] and Tolwinski [20], have examined the nonlinear multilevel formulation for optimal control problems with strictly convex quadratic cost functions and linear equality constraints. Finally, Luh et al. [16] address the case where all the decision variables are discrete.

To solve (1) with $n_2 > 0$, we begin by replacing the follower's optimization problem with appropriate Kuhn–Tucker conditions. This leads to

$$\underset{x, w, y, \mu}{\text{minimize}} \quad F(x, w, y) \tag{2a}$$

$$\text{subject to} \quad \nabla_y f(x, w, y) + \mu \nabla_y g(x, w, y) = 0, \tag{2b}$$

$$g(x, w, y) \leq 0, \tag{2c}$$

$$\mu \geq 0, \tag{2d}$$

$$\mu g(x, w, y) = 0, \tag{2e}$$

$$w_l \in \{0, 1\}, \qquad l = 1, \ldots, n_2, \tag{2f}$$

where μ is an m-dimensional vector of Lagrange multipliers.

In the sequel, we assume F is convex, f is a convex quadratic function, and all the elements of g are linear. This implies that (2b) is linear, and that a global solution of (2) likewise solves (1) [11,18]. However, (2) is a nonconvex program and may have saddle points and local optima. We circumvent this difficulty by relaxing the complicating constraints (2e) and integrality requirements (2f) in order to obtain a convex program in continuous variables. Branch and bound is then used to enforce feasibility.

In particular, consider a relaxation of (2) obtained by eliminating the complementary slackness conditions (2e), and replacing the integrality requirements (2f) by the bounds $0 \leq w_l \leq 1$, $l = 1, \ldots, n_2$. This results in an easy-to-solve convex program. Feasibility with respect to complementary slackness conditions and integrality requirements on w can then be achieved by selectively introducing constraints $\mu_i = 0$ or $g_i = 0$, and $w_l = 0$ or $w_l = 1$ to the formulation.

3. Branch and bound algorithm

In a previous work, Edmunds [11] solved (1) with only continuous variables using branch and bound to satisfy the complementary slackness conditions (2e). We now extend this approach by including integer variables in the leader's problem and considering constraints of the form $w_l = 0$ or $w_l = 1$ in the search tree.

When the convex program described in section 2 is solved, some of the relaxed constraints may be violated. In our procedure, one of the violated complementary slackness conditions or integrality requirements is selected and two subproblems are set up. If the ith complementary slackness condition is selected, the first subproblem corresponds to the case where $\mu_i = 0$ and second to $g_i = 0$. Similarly, if the lth integrality requirement is selected, the first subproblem corresponds to the case where $w_l = 0$ and the second to $w_l = 1$. Solutions to the two subproblems are obtained and the procedure is repeated. The search continues until all of the complementary slackness

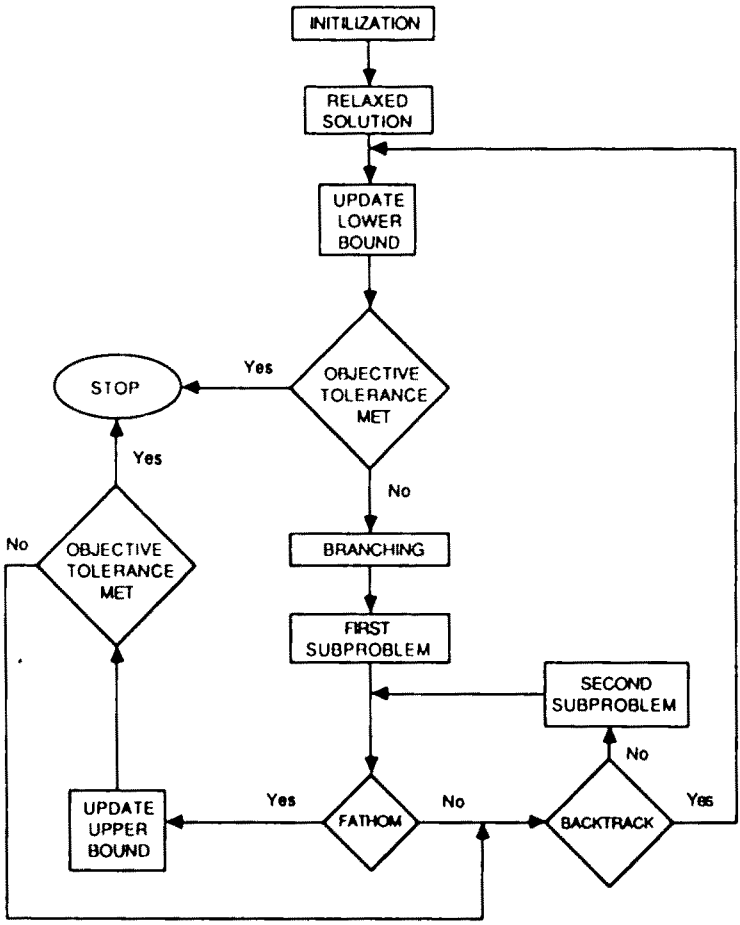

Fig. 1. Branch and bound algorithm.

conditions (2e) and integrality requirements (2f) are satisfied. The algorithm terminates when either all subproblems have been solved, are known to have solutions that are suboptimal, or are infeasible. This modified breadth-first branch and bound search is implemented in the algorithm described below. Alternative search techniques are discussed in section 4. A flowchart of the algorithm is displayed in fig. 1.

In the following discussion, we denote the incumbent lower and upper bounds on the leader objective function as $\underline{F}$ and $\bar{F}$, respectively. A live node refers to a subproblem solution which violates one or more of conditions (2e) and (2f), and has an objective function value lower than $\bar{F}$. The variable j indexes the set of live nodes, while k is an iteration counter. Finally, let $D = \{1, 2, \ldots, m, m+1, m+2, \cdots, m+n_2\}$ be the index set for the m complementary slackness conditions (2e) and the n_2 integrality conditions (2f).

Step 1: (Initialization) Set $\bar{F} = \infty$ and $k = 0$.

Step 2: (Relaxed solution) Solve (2) with (2e) relaxed and (2f) replaced by $0 \le w_l \le 1$ for all l. If (2e) and (2f) are satisfied, label the solution (x^*, w^*, y^*) and set $\bar{F} = F(x^*, w^*, y^*)$; otherwise, this subproblem solution corresponds to a live node.

Step 3: (Update lower bound) Let j' index the live node with minimum objective function value. Set $\underline{F}$ = objective function value for subproblem j'.

Step 4: (Objective tolerance) If $[\bar{F} - \underline{F}]$ is within tolerance, terminate with an ε-optimal solution (x^*, w^*, y^*).

Step 5: (Branching) For the solution corresponding to live node j', choose the index in the set D corresponding to

$$\max\left(|\mu_l^{j'} g_l^{j'}|, \min\left(w_l^{j'}, \left(1 - w_l^{j'}\right)\right)\right)$$

and label the index i'.

Step 6: (First subproblem) Add one of the two constraints which may be used to enforce the condition associated with index i' and solve.

Step 7: (Fathom) If the subproblem is infeasible or $F(x^k, w^k, y^k) > \bar{F}$, go to step 10; otherwise, label solution (x^k, w^k, y^k, μ^k) and add this subproblem solution to the set of live nodes.

Step 8: (Update upper bound) If (2e) and (2f) are satisfied at (x^k, w^k, y^k), then set $(x^*, w^*, y^*) \leftarrow (x^k, w^k, y^k)$, $\bar{F} = F(x^k, w^k, y^k)$ and remove the subproblem solution from the set of live nodes.

Step 9: (Objective tolerance) If $[\bar{F} - \underline{F}]$ is within tolerance, then terminate with ε-optimal solution (x^*, w^*, y^*).

Step 10: (Backtrack) If both subproblems associated with condition i' have been examined, then go to step 3.

Step 11: (Second subproblem) Remove the first constraint added at step 6 and append the second constraint associated with condition i'. Solve subproblem, set $k \leftarrow k + 1$ and go to step 7.

In step 1, the upper bound is set to infinity and the iteration counter is set to zero. A relaxed version of (2) is solved at step 2, and the upper bound is updated if this solution happens to satisfy constraints (2e) and (2f). At step 3, the lower bound is set equal to the minimum objective function value of all live nodes, and the upper and lower bounds are compared in step 4.

A violated complementary slackness condition or integrality requirement is selected in step 5 to be satisfied by adding a constraint in step 6. In step 7, this new subproblem is fathomed if it is infeasible or its solution is worse than the incumbent upper bound. In step 8, the upper bound is updated when appropriate and compared with the lower bound in step 9.

If both subproblems associated with index i' have been solved, step 10 returns control to step 3, where a new live node is selected for further exploration. Otherwise, the second subproblem is formulated and solved in step 11.

4. Computational experience

The basic algorithm has been coded in VS FORTRAN to run on an IBM 3081-D mainframe. A successive quadratic programming (SQP) package is used to solve the subproblems at steps 2, 6 and 11 (see Fan et al. [12]). The purpose of the following numerical tests is to identify the most promising branching strategy for implementation in step 5. This is done by comparing a variety of breadth-first and depth-first search techniques.

The randomly generated problems used for this testing are characterized by convex quadratic objective functions and linear constraints (see Edmunds [11] for a discussion of the generation procedure). As indicated in section 2, the QPs associated with the relaxation of these problems are convex, implying that global optima can be obtained at each node in the branch and bound tree.

The breadth-first search technique outlined in section 3 was the one ultimately chosen for implementation. One version of this approach that works well for standard mixed-integer linear programs with zero-one variables [19] is to select one or more violated conditions in step 5 and generate the associated subproblems. For example, if two violated conditions at node k are selected, each of the subproblems derived would incorporate two additional constraints. Hence, there would be a total of four subproblems associated with node k which would be generated and solved. We use the parameter η to refer to the number of violated conditions that are selected to form related subproblems. The cases where $\eta = 1$ and 2 are examined below.

A depth-first search technique may also be used to explore the branch and bound tree. Here, the most recently generated node is selected for further exploration unless the problem is infeasible or the objective function value is greater than the upper bound.

Branching rules determine which constraint is to be added to form the next subproblem in the depth-first search, and greatly affect the efficiency of the approach

(see Bard and Moore [5]). If violated complementary slackness condition i is selected according to step 5, one branching rule may require that the constraint $\mu_i = 0$ be added to form the next subproblem, while another rule may require the addition of $g_i = 0$. The constraint not selected is added later during the backtracking operation. We examine the following three branching rules.

RULE 1

If a complementary slackness condition has been selected in step 5, include constraint $\mu_i = 0$. If an integrality requirement has been selected, include constraint $w_l = 0$.

RULE 2

If a complementary slackness condition has been selected in step 5, include constraint $g_i = 0$. If an integrality requirement has been selected, include constraint $w_l = 1$.

RULE 3

If a complementary slackness condition has been selected in step 5, and $\mu_i < -g_i$, include $\mu_i = 0$; otherwise, include $g_i = 0$. If an integrality requirement has been selected and $w_l < 1 - w_l$, include constraint $w_l = 0$; alternatively, include $w_l = 1$.

Note that rule 3 selects the constraint that is most nearly satisfied by the current solution. The idea is that the addition of this constraint will introduce the smallest possible perturbation into the problem.

In tables 1 through 5, ten test problems involving 2 continuous leader variables ($n_1 = 2$), 3 binary leader variables ($n_2 = 3$), 5 follower variables ($n_3 = 5$), and 5 inequality constraints ($m = 5$) are solved. All continuous variables are free. Each table reports results using a different search technique. In all, we investigate breadth-first search with $\eta = 1$ and $\eta = 2$, and depth-first search using branching rules 1, 2 and 3. Output statistics include CPU time (seconds), the total number of nodes in the branch and bound tree, and the node at which the optimum is found. The column labeled "Funct. calls" indicates the number of times each function in the problem is evaluated. The column labeled "Deriv. calls" indicates the number of times the partial derivative of each function is evaluated. Table 6 provides a summary of the mean values reported in tables 1 through 5.

It is clear from the data in table 6 that the breadth-first search technique with $\eta = 1$ provides superior performance with regard to computation time. As opposed to depth-first search, this technique offers the flexibility to explore the most promising nodes first rather than always branching on the most recently generated candidate. Hence, good upper bounds are found quickly and large portions of the tree can be eliminated from consideration.

Table 1

Breadth-first search with $\eta = 1$

Prob.	Virtual CPU sec	Total nodes	Opt. node	Funct. calls	Deriv. calls
1	3.2	5	5	108	1150
2	7.2	13	8	368	2575
3	6.0	7	7	137	1575
4	5.0	7	6	213	1675
5	5.1	9	6	306	1900
6	11.8	15	13	349	3250
7	9.5	15	11	413	2950
8	3.0	3	2	82	800
9	22.9	9	9	598	4100
10	7.7	7	7	142	1675
Mean:	8.1	9	7	271	2165

Table 2

Breadth-first search with $\eta = 2$

Prob.	Virtual CPU sec	Total nodes	Opt. node	Funct. calls	Deriv. calls
1	3.2	5	5	108	1150
2	8.0	17	6	644	2925
3	7.2	9	7	270	2125
4	5.2	7	6	179	1775
5	3.6	7	4	177	1400
6	11.8	15	6	303	3300
7	9.0	15	11	397	2850
8	3.0	3	2	82	800
9	40.2	13	13	1258	6100
10	7.9	7	7	158	1775
Mean:	9.9	10	7	358	2420

Table 3

Depth-first search with rule 1

Prob.	Virtual CPU sec	Total nodes	Opt. node	Funct. calls	Deriv. calls
1	5.1	9	9	254	1800
2	6.6	13	5	402	2425
3	12.0	17	15	550	3675
4	88.9	15	13	4745	13325
5	4.6	9	8	212	1825
6	18.1	19	9	426	3925
7	11.3	17	8	545	3350
8	2.9	3	2	81	800
9	43.1	21	18	1211	7500
10	23.2	33	32	762	6825
Mean:	21.6	16	12	919	4545

Table 4

Depth-first search with rule 2

Prob.	Virtual CPU sec	Total nodes	Opt. node	Funct. calls	Deriv. calls
1	3.2	5	3	147	1100
2	8.5	17	16	654	3025
3	151.0	27	26	1825	7675
4	8.5	13	12	421	2825
5	4.9	9	4	260	1825
6	40.1	19	18	1792	7025
7	10.0	15	14	429	3050
8	7.7	15	15	442	2875
9	18.3	11	9	531	3925
10	38.4	11	9	760	4025
Mean:	29.1	14	13	726	3735

Table 5

Depth-first search with rule 3

Prob.	Virtual CPU sec	Total nodes	Opt. node	Funct. calls	Deriv. calls
1	4.3	9	8	199	1700
2	6.7	13	5	402	2425
3	6.4	7	4	186	1775
4	89.0	15	13	4745	13325
5	4.8	9	8	212	1825
6	15.1	15	4	295	3300
7	10.0	15	5	396	2950
8	2.9	3	2	81	800
9	43.5	21	17	1208	7375
10	43.8	27	26	1487	8300
Mean:	22.7	13	9	921	4378

Table 6

Comparison of techniques

Strategy:	Mean virtual CPU sec	Mean total nodes	Mean opt. node	Mean funct. calls	Mean deriv. calls
$\eta = 1$ (breadth-first)	8.1	9	7	271	2165
$\eta = 2$ (breadth-first)	9.9	10	7	358	2420
Rule 1 (depth-first)	21.6	16	12	919	4545
Rule 2 (depth-first)	29.1	14	13	726	3735
Rule 3 (depth-first)	22.7	13	9	921	4378

We now investigate some larger problems using this strategy. In table 7, results from test problems with 5 continuous leader variables, 5 binary leader variables, 10 follower variables, and 10 inequality constraints are reported. The wide variation in computation time is due to the variation in the number of nodes examimed.

Table 7

Breadth-first search, problem set 1, $\eta = 1$ ($n_1 = 5$, $n_2 = 5$, $n_3 = 10$, $m = 10$)

Prob.	Virtual CPU sec	Total nodes	Opt. node	Funct. calls	Deriv. calls
1	46.3	7	7	270	4000
2	142.4	33	32	679	15050
3	147.7	15	8	329	7350
4	109.4	21	12	479	10400
5	406.1	71	16	1653	35350
Mean:	170.4	29	15	686	14430

Table 8

Breadth-first search, problem set 2, $\eta = 1$ ($n_1 = 7$, $n_2 = 7$, $n_3 = 12$, $m = 12$)

Prob.	Virtual CPU sec	Total nodes	Opt. node	Funct. calls	Deriv. calls
1	573.5	17	17	492	13702
2	278.9	9	9	223	6138
3	606.1	23	22	549	15066
4	465.6	19	19	454	12586
5[a]	900.0	-	-	-	-
Mean[b]:	481.0	17	17	430	11837

[a] Solution not found in 900 CPU seconds.
[b] Mean excluding problem 5.

Finally, results from a third problem set are shown table 8. These problems involve 7 continuous leader variables, 7 binary leader variables, 12 follower variables, and 12 inequality constraints. We have added two more binary variables and two more constraints, thus increasing the size of the corresponding tree by a factor of sixteen. The results indicate an approximately threefold increase in computational effort, indicating that the algorithm is not necessarily exponential. Nevertheless, as n_2 and m grow, it is less likely that we will be able to obtain optimal solutions in a reasonable amount of time. Given the inherent difficulty of the problem, larger-scale instances may prove to be totally intractable.

Note the wide variation in computation times. As in most branch and bound approaches, there is an element of chance involved in the success of the search

procedure. If there are many promising branches in the tree which must be explored, then the procedure will be time consuming. In such cases, a tradeoff of computation time for an ϵ-optimal solution may be a prudent choice.

5. Integer follower decision variables

In this section, we consider BLPPs in which some of the follower's decision variables are discrete. The purpose is to show through illustration that it is not possible in general to extend the above ideas to this case. This result is to be expected because the follower's Kuhn–Tucker conditions identify the optimum only for continuous functions. Discrete variables in the follower's problem introduce discontinuities that preclude the use of Kuhn–Tucker conditions. Similar findings are reported by Moore and Bard [17] for the linear BLPP.

In the first example, we demonstrate the fact that the relaxed problem may not provide a valid lower bound on (1). Toward this end, consider the following BLPP:

$$\underset{x}{\text{minimize}} \quad F = (x-2)^2 + (y-2)^2 \qquad \text{where } y \text{ solves} \tag{3a}$$

$$\underset{y}{\text{minimize}} \quad f = y^2 \tag{3b}$$

$$\text{subject to} \quad 2x + 2y \geq 5, \tag{3c}$$

$$x - y \leq 1, \tag{3d}$$

$$3x + 2y \leq 8, \tag{3e}$$

$$y \in \{0, 1, 2\}, \tag{3f}$$

where x is a continuous variable under the leader's control, and y is restricted by (3f) to take on one of three values.

In this problem, the leader attempts to attain a point as close as possible to (2, 2) by selecting a permissible value of x in accordance with constraints (3c), (3d) and (3e). The follower observes this value and chooses the smallest feasible integer for y. The geometry of this problem is illustrated in fig. 2.

Now consider problem (3) with (3f) relaxed. The set of attainable points for the leader is shown by the darkened line in fig. 2, and is known as the inducible region. The continuous solution for this problem is point A, where $x = 2$, $y = 1$, and $F(2, 1) = 1$. Because these values satisfy (3f), we might conclude that a valid lower bound to (3) is $F = 1$. However, this conclusion is incorrect. If the leader chooses $x = 4/3$, the follower must choose the integer $y = 2$ and we attain the point B, where $F(4/3, 2) = 4/9$. Hence, the solution to the relaxed problem is worse than the solution to the original problem. This implies that the branch and bound procedure of section 3 is no longer valid when some of the follower's variables have integer restriction.

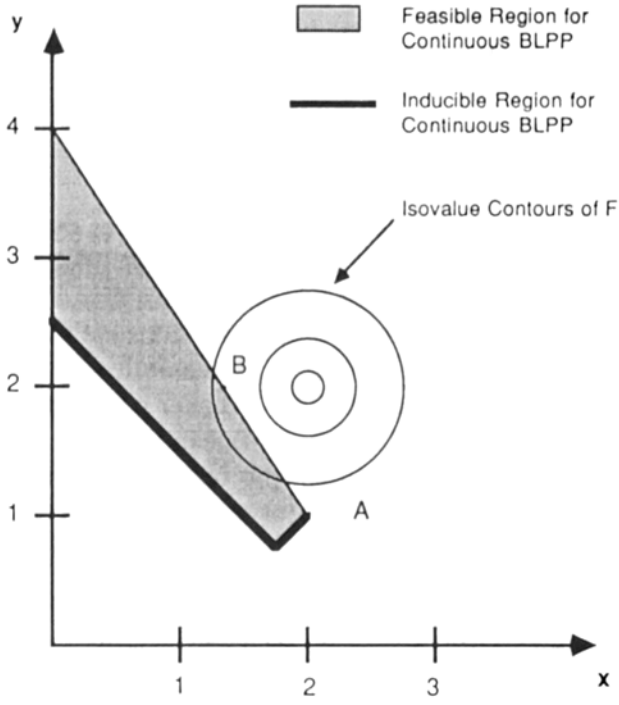

Fig. 2. Geometry for problem (3).

The second example illustrates a situation in which the minimum of $F(x, y)$ does not exist. Consider the following BLPP:

$$\underset{x}{\text{minimize}} \quad F = (x-2)^2 + (y-2)^2 \qquad \text{where } y \text{ solves} \tag{4a}$$

$$\underset{y}{\text{minimize}} \quad f = y^2 \tag{4b}$$

$$\text{subject to} \quad 2x + 3y \geq 6, \tag{4c}$$

$$x \leq 2, \tag{4d}$$

$$y \in \{0, 1, 2\}, \tag{4e}$$

where x is a continuous variable under the leader's control.

In this example, the leader attempts to induce the follower to choose $y = 2$ while making x as large as possible. The constraint (4c) aids the leader by forcing $y = 2$ if $x < 1.5$. However, at $x = 1.5$, the follower changes his choice to $y = 1$. Thus, the leader can never attain the value $F(1.5, 2) = 0.25$ but can get arbitrarily close; i.e. $F(1.5 - \varepsilon, 2) = 0.25 - \varepsilon + \varepsilon^2$, where ε is an arbitrarily small constant. Hence, $\min\{F(x, y)\}$ does not exist for (4). This situation is illustrated in fig. 3.

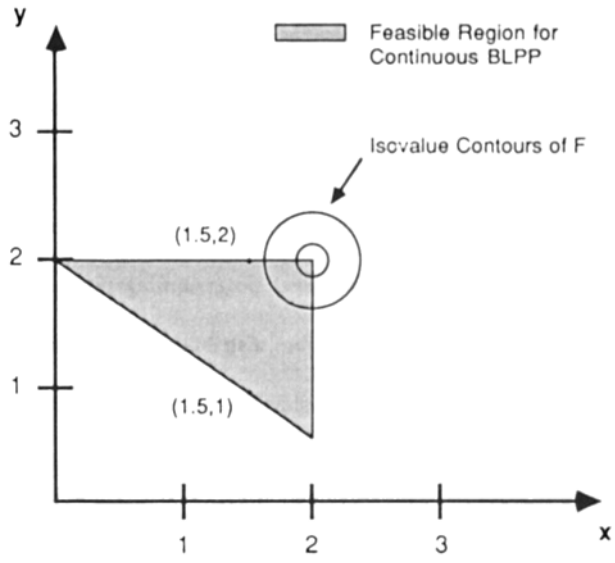

Fig. 3. Geometry for problem (4).

5. Summary and conclusions

The main purpose of this paper has been to present a branch and bound algorithm that will solve linear–quadratic BLPPs containing integer decision variables under the leader's control. Breadth-first search techniques appear to be more efficient than depth-first search techniques in solving such problems. Test results confirm that setting up and solving two new subproblems at each iteration gives the best performance. More research is needed, though, to find ways of cutting down the overall computational effort.

Two examples are included in the final section to underscore some of the difficulties associated with solving BLPPs in which both players have control of integer decision variables. One example illustrates the problem of finding valid lower bounds, while the other demonstrates that the minimum of the leader's objective function may not always exist. Together, these difficulties indicate that the branch and bound techniques developed in section 3 are not guaranteed to solve BLPPs in which the follower controls integer decision variables.

References

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

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

[3] J.F. Bard, Coordination of a multidivisional firm through two levels of management, Omega 11(1983)457–468.

[4] J.F. Bard, Convex two-level optimization, Math. Progr. 40(1988)15–27.
[5] J.F. Bard and J.T. Moore, A branch and bound algorithm for the linear bilevel programming problem, SIAM J. Sci. Statist. Comput. 11(1990)281–292.
[6] J.F. Bard and J.E. Falk, An explicit solution to the multi-level programming problem, Comput. Oper. Res. 9(1982)77–100.
[7] T. Basar and H. Selbuz, Closed loop Stackelberg strategies with applications in optimal control of multilevel systems, IEEE Trans. Auto. Control AC-24(1979)166–178.
[8] W.F. Bialas and M.H. Karwan, On two-level optimization, IEEE Trans. Auto. Control AC-27(1982)211–214.
[9] W. Candler and R. Townsley, A linear two-level programming problem, Comput. Oper. Res. 9(1982)59–76.
[10] R. Cassidy, M.J. Kirby and W.M. Raike, Efficient distribution of resources through three levels of government, Manag. Sci. 17(1971)462–473.
[11] T.A. Edmunds, Algorithms for nonlinear bilevel mathematical programs, Ph.D. Dissertation, Department of Mechanical Engineering, University of Texas at Austin (1988).
[12] Y. Fan, S. Sarkar and L. Lasdon, Experiments with successive quadratic programming algorithms, Working Paper, Department of General Business, University of Texas, Austin (1985).
[13] J. Fortuny-Amat and B. McCarl, A representation and economic interpretation of a two-level programming problem, J. Oper. Res. Soc. 32(1981)783–792.
[14] P. Hansen, B. Jaumard and G. Savard, A variable elimination algorithm for bilevel linear programming, RUTCOR Research Report RRR 17-89, Rutgers University (1989).
[15] J. Júdice and A. Faustino, The solution of the linear bilevel programming problem by using the linear complementarity problem, Investigacao Operacional 8(1988)77–95.
[16] P.B. Luh, T.S. Chang and T.K. Ning, Three-level hierarchical decision problems, *Proc. 5th MIT/ONR Workshop on Command, Control, and Communications Systems*, Monterey, CA (1982).
[17] J.T. Moore and J.F. Bard, The mixed integer bilevel programming problem, Oper. Res. 38(1990) 911–921.
[18] M. Simaan and J.B. Cruz, Jr., On the Stackelberg strategy in nonzero-sum games, J. Optim. Theory Appl. 11(1973)533–555.
[19] J. Singhal, R.E. Marsten and T.L. Morin, Fixed order branch-and-bound methods for mixed-integer programming: The ZOOM system, ORSA J. Comput. 1(1989)44–51.
[20] B. Tolwinski, Closed-loop Stackelberg solution to a multi-stage linear–quadratic game, J. Optim. Theory Appl. 34(1981)485–501.