Comput. & Ops Res. Vol. 9, No. 1, pp. 77–100, 1982
Printed in Great Britain.

0305-0548/82/010077-24$03.00/0
Pergamon Press Ltd.

AN EXPLICIT SOLUTION TO THE MULTI-LEVEL PROGRAMMING PROBLEM†

JONATHAN F. BARD‡ and JAMES E. FALK§

Institute for Management Science and Engineering, School of Engineering and Applied Science, The George Washington University, Washington, DC 20052, U.S.A.

Abstract—The multi-level programming problem is defined as an n-person nonzero-sum game with perfect information in which the players move sequentially. The bi-level linear case is addressed in detail. Solutions are obtained by recasting this problem as a standard mathematical probram and appealing to its implicitly separable structure. The reformulated optimization problem is linear save for a complementarity constraint of the form $\langle u, g\rangle = 0$. This constraint is decomposed in a manner that permits us to achieve separability with very little cost in dimensionality. A general branch and bound algorithm is then applied to obtain solutions. Unlike the conventional mathematical program though, the multi-level program may fail to have a solution even when the decision variables are defined over a compact set. An auxiliary optimization problem is employed to detect such failure. Finally, the general max–min problem is discussed within the bi-level programming framework. Examples are given for a variety of related problems.

1. INTRODUCTION

Multi-level programming was first defined by Candler and Townsley[12] as a generalization of mathematical programming. In this context, the constraint region is implicitly determined by a series of optimization problems which must be solved in a predetermined sequence (see Bracken and McGill[8]). Alternatively, the problem can be viewed as an n-person, nonzero-sum game with perfect information (Luce and Raiffa[28]) where the order of play is specified at the outset and the players' strategy sets are no longer assumed to be disjoint. As a consequence, the moves available to a player change as the game progresses and hence, may be limited by the actions of the preceding players. When interdependent strategy sets are introduced the difficulty of the overall problem markedly increases.

The problem that we address differs from the conventional formulation of the n-person game in that our players are required to move in turn. When the moves are assumed to occur simultaneously, disagreement often arises as to which of several measures is most likely to predict the actual outcome (see Davis and Maschler[14] or Luce and Raiffa[28], Chapter 9). We avoid such arguments by appealing to the natural relationship between the multi-level program and the standard mathematical program and define a solution accordingly. This leaves us free to focus on the problematic nature of the computations.

To define the problem, suppose there are n optimizers, each of whom wishes to maximize his own objective function f_i. As in the conventional setting, each optimizer has control over a set $X_i \subset R^{n^1}$ of decision variables, and each objective function collectively depends on the decisions made by each optimizer. Here, however, we shall assume that optimizer 1 has the first choice and selects $x^1 \in X^1$, followed by optimizer 2 who selects $x^2 \in X^2$, and so on through optimizer n.

We shall further assume that the choices made by optimizers 1 through i may affect the set of feasible strategies available to optimize $i+1$. Now let $g^i : R^{n^1+\cdots+n^i} \to R^{m^i}$, $i = 1, \ldots, n$, be given functions of $x^1, \ldots, x^i$; then the set

$$\{x^i : g^i(x^1, \ldots, x^{i-1}, x^i) \geqq 0\} \cap X^i$$

†Supported by Office of Naval Research, Program in Logistics, Contract N00014-75-C-0729, Project NR 347 020.

‡Dr. Bard is Assistant Professor of Management Science at the University of Massachusetts. He holds the D.Sc. in operations research from The George Washington University (1979), the MS in systems engineering from Stanford University (1969), and the BS from Renssalear in aerospace engineering (1968). He has been employed by the Aerospace Corporation, Booz Allen and Hamilton, and the MITRE Corporation.

§Dr. Falk is one of the special editors of this issue. His qualifications are noted in the Preface.

depends on the settings of x^1 through x^{i-1}. For simplicity, we shall assume these sets are not empty; i.e. the ith player always has some recourse.

Optimizer 1's problem then becomes

$$\left.\begin{array}{ll} \max f_1(x^1, x^2, \ldots, x^n) & \text{where } x^2 \text{ solves} \\ x^1 \in X^1 & \\ g^1(x^1) \geqq 0 & \\ \max f_2(x^1, x^2, \ldots, x^n) & \text{where } x^3 \text{ solves} \\ x^2 \in X^2 & \\ g^2(x^1, x^2) \geqq 0 & \\ \quad \cdots & \\ \max f_n(x^1, x^2, \ldots, x^n) & \text{where } x^n \text{ solves} \\ x^n \in X^n & \\ g^n(x^1, \ldots, x^n) \geqq 0 & \end{array}\right\} \tag{1}$$

If $n = 1$, problem (1) becomes a standard optimization problem. If $n = 2$, and $f_2(x) = -f_1(x)$, problem (1) is equivalent to the "max–min" problem

$$\max_{\substack{x^1 \in X^1 \\ g^1(x^1) \geqq 0}} \min_{\substack{x^2 \in X^2 \\ g^2(x^1, x^2) \geqq 0}} f_1(x)$$

Even here, note that this is not a standard max–min problem due to the dependency of the inside optimizer's constraints on x^1. Indeed, in both of these cases, optimal solutions are guaranteed to exist when f_1, g^1, g^2 are continuous over the compact sets X^1 and X^2. This is not true in general for the multi-level program, as we shall see in Section 3.1.

In this paper, we will restrict ourselves to the case where $n = 2$ and all the functions are linear. Even with these restrictions, the corresponding problem is equivalent to a nonconvex program and thus can have local optima. As such, little hope exists in developing an algorithm that would exhibit monotonic improvement in the objective function. The approach that we use is based on a transformation of the two-level or bi-level programming problem into a standard mathematical program. The resulting optimization problem, distinguished by a complementarity constraint, is decomposed in a manner similar to that proposed by Bard and Falk[4] for computing equilibria. A branch and bound algorithm is applied to obtain solutions. The algorithm itself is applicable to any nonconvex program whose functions are upper semicontinuous and can be put into a separable form. Consequently, extensions of the proposed method are limited to cases where the objective functions f_1 and f_2 in (1) are nonlinear and separable.

An economic interpretation of the bi-level programming problem has been offered by Candler and Townsley. The first player (or outside player) is referred to as the higher level decision maker who has control over "policy variables," x^1. For example, this player or team of players may be able to control directly tax rates and the size of the government's budgetary deficit, thus setting policy. The second player (or inside player) is referred to as the lower level decision maker who has control of the (remaining) "behavioral variables", x^2, which are manipulated in light of the levels of the policy variables. The behavioral variables such as rate of private investment, and agricultural production may be decided by many decentralized decision makers following their own behavioral rules, but the effect is viewed collectively.

In addition, a third level can be added if the higher level decision maker wishes to influence a third set of "impact" variables, generally outside the control of either decision making group. Although no direct means of control may exist for such variables as pollution, unemployment, rate of inflation, and balance of payments, policy makers most certainly would like to influence their impact on the economy.

Throughout this paper we will find it convenient to substitute the vector (x, y) for (x^1, x^2) when only two players are involved. In the next section we discuss a variety of related problems and the Candler–Townsley approach to the two-level linear programming problem. Next, some structural considerations relating to the absence of solutions and the order of play are presented. This is followed by the development of an alternative solution technique based

on nonconvex programming. Examples are given to demonstrate the computational aspects of the approach. Finally, the max–min problem is explored in the current context.

2. BACKGROUND

In spite of its potential applicability, the multi-level programming problem has been given little attention and is only now beginning to emerge as an independent component of nonconvex programming. Problems of this type have recently been investigated in connection with government oil pricing schemes (De Silva[15]) and armed conflict (Bracken *et al.*[9]). In addition, a host of related examples can be found in the well-established area of n-person game theory (see e.g. Owen[32]). In order to compare the structural similarities and differences between these two areas, we present an outline of the n-person game in normal form. This game consists of: (i) a set of n players; (ii) the n strategy sets $S_1, S_2, \ldots, S_n$; and (iii) the n real-valued payoff functions $f_1, f_2, \ldots, f_n$ where $f_i(x^1, \ldots, x^n)$ is the payoff to player i when the players j $(j = 2, \ldots, n)$ use strategies $x^i \in S_i$. A basic assumption is that each player knows the entire structure of the game in this form and that all players are governed in their behavior by an inflexible desire to maximize their expected payoff (Luce and Raiffa[28]).

In contrast to (1) it is usually assumed that the strategy sets S_i for all i are independent or disjoint and that all players move simultaneously. A further consideration is one of cooperation among the players. While this may work out to everyone's advantage, instances arise where the rules of the game or the realities of the situation strictly forbid any type of agreements (e.g. anti-trust laws or the inability to communicate). Two cases must therefore be distinguished:

(1) The noncooperative case, in which any type of collision, such as correlated strategies and side payments, is prohibited.

(2) The cooperative case, in which all such agreements are permitted.

The noncooperative case most accurately reflects the assumptions implicit in the multi-level programming problem.

In the remainder of this section, we will highlight the structure and properties of a special n-person game known as the bimatrix game. This will be followed by a discussion of a variant of the standard mathematical program which contains optimization problems in the constraints. Such problems stand mid-way between the multi-level and standard form of the mathematical program, while containing elements of the two-person, zero-sum game. We will close by sketching the approach used by Candler and Townsley in solving the two-level linear program.

2.1 *The bimatrix game*

In general, a finite two-person nonzero-sum game can be expressed as a pair of $m \times n$ matrices, $A = (a_{ij})$ and $B = (b_{ij})$, or equivalently, as an $m \times n$ matrix (A, B) each of whose entries is an ordered pair (a_{ij}, b_{ij}). The entries a_{ij} and b_{ij} are the payoffs (in utilities) to the players I and II, assuming they choose, respectively, their ith and jth pure strategies. A game in this form is called a bimatrix game.

Definition 1. A mixed strategy for I is a column x of nonnegative elements x_i, which represent the probability with which I will play his ith pure strategy. Thus $x_1 + x_2 \cdots + x_m = 1$. Likewise, a mixed strategy for II is a column y whose nonnegative components y_j sum to 1.

If on each play of the game I and II select a pure strategy randomly, according to the probability distributions given by x and y, their expected payoffs in matrix form are

$$\langle x, Ay \rangle \quad \text{and} \quad \langle x, By \rangle.$$

A "solution" to the game is often characterized by an equilibrium point—a collection of strategies, one for either player, such that no player is able to increase his payoff by changing his strategy choice when the other holds his fixed. More formally, an equilibrium pair is defined as follows.

Definition 2. A pair of mixed strategies (x^*, y^*) for the bimatrix game (A, B) is said to be in equilibrium if, for any other mixed strategies, x and y

$$\langle x, Ay^* \rangle \leqq \langle x^*, Ay^* \rangle$$

$$\langle x^*, By \rangle \leqq \langle x^*, By^* \rangle.$$

Nash[31] has shown by a fixed point argument that every game has at least one mixed strategy equilibrium pair. In general, equilibrium points are neither equivalent (yield the same payoffs) nor interchangeable (yield an equilibrium point when such strategies are intermixed). It has long been known (see, e.g. Tucker[34]) that an equilibrium pair for the zero-sum matrix game, i.e. the case where $B = -A$, may be recognized as a pair of optimal solutions to an associated dual pair of linear programs. Although no direct linear formulation exists in the general case, Lemke and Howson[27] have shown by an algebraic argument, that an equilibrium pair lies on a path joining a sequence of adjacent extreme points of a certain convex polyhedron. Such a path, and hence equilibrium point, is readily computed by exploiting common linear programming techniques; namely, the perturbation of a convex polyhedron, and the generation of adjacent extreme points. The constructive proofs presented in their paper demonstrate the equivalence of the bimatrix game and the linear complementarity problem (Eaves[16], Mangasarian[29]), which is further explored in a sequel by Lemke[29]. His results apply to the case where the matrices A and B exhibit special structures; Bard and Falk[4] offer a full treatment of the general case.

2.2 *Optimization problems in the constraints*

The second type of problem that we will discuss also has a game-theoretic interpretation which can be extended to any number of players acting in sequence. Its structure derives from the standard form of the mathematical program as parameterized by the vector $t \in T \subset R^p$. The associated problem is one of finding a vector $x \in R^n$ to

$$\max \{(f(x): x \in S(t)\} \tag{2}$$

where

$$S(t) = \{x: g(x, t) \leqq 0\} \tag{3}$$

$f: R^n \to R$ and $g: R^n \to R^m$. This problem has been studied extensively from the point of view of sensitivity analysis (e.g. see Fiacco and McCormick[22] or Armacost and Fiacco[1]). As t varies over a set of values T, the minimal value of the objective function may also vary. Evans and Gould[17] give conditions for which this variation is a continuous function of t. Fiacco[21] extends this work to establish a theoretical basis for utilizing a penalty function method to extimate sensitivity information of a local solution and its associated Lagrange multipliers.

If (2) must be satisfied for all values of $t \in T$, we get the infinitely constrained problem (Blankenship and Falk[6]). If, however, the parameter t is viewed as an additional variable subject to the control of a second decision maker, we get an optimization problem within an optimization problem. To see this, consider the problem proposed by Bracken and McGill[8] of finding vectors x and t to

$$\max_{x \in X} f(x) \tag{4.1}$$

subject to

$$h(x) = \max_{t \in T} g(x, t) \leqq 0 \tag{4.2}$$

where $h: R^n \to R^m$ and X is a compact of R^n. Problem (4) differs slightly from problem (1) in that X and T are independent, f is not a function of t, and the outside player's selection, x^* must be such that the maximum value of $g(x^*, t)$ does not exceed zero. Further, the inside player now has m rather than one function to optimize. In both cases, it is assumed that all functions and constraint sets are generally known.

Alternatively, rather than viewing the constraint region (4.2) to be under the control of one player, we may assign the control of each of the m constraints to a different player. From a structural point of view, this interpretation, though suggesting a much more complex problem, would bring (4) closer in line with (1).

It is possible to generalize the constraint region (4.2) to allow the solution of problem (4) to be constrained by the value of a two-sided optimization problem. Bracken and McGill indicate two ways in which the new feasible region may be parameterized by x, the primary decision variable. To achieve the desired formulation both the objective function and the constraint set of the outside player associated with the two-sided problem will be defined as a function of x. The following problem illustrates this parameterization: find vectors $x = (x_1, \ldots, x_n)$, $t = (t, \ldots, t_p)$, and $u = (u_1, \ldots, u_r)$ to

$$\max_{x \in X} f(x) \tag{5.1}$$

subject to

$$\bar{h}(x) = \min_{u \in U(x)} \max_{t \in T} g(x, u, t) \leqq 0 \tag{5.2}$$

where $U(x) \subset R^r$ is assumed to have the following form:

$$U(x) = \{u: d_i(x, u) \leqq 0, \quad 1 \equiv 1, 2, \ldots, \bar{m}\}$$

which is analagous to the set parameterization given in (3).

In order to interpret (5), note if the two-sided optimization problem given in (5.2) has a saddle-point in (u, t), then the solution of this two-sided problem does not depend on the order in which u and t are chosen. Thus, x is chosen first followed by choices of u and t. If there is not a saddle-point in (u, t), then the proper interpretation is that of the outside player choosing x and u, followed by the inside player choosing t. In this case, the two-sided problem in constraint i in (5.2) is a min–max problem.

Conditions are given in [8] for which problems (4) and (5) are convex programs—a desirable feature if the work required to obtain global solutions is to be minimized. A computer program called INSUMT has been developed[7] for use with the sequential unconstrained minimization technique (SUMT) of Fiacco and McCormick[22] to solve mathematical programs with optimization problems in the constraints.

2.3 *The Candler–Townsley approach to the bi-level linear program*

The method developed in [12] for solving the bi-level linear program is based on an implicit enumeration scheme which generates global information at each iteration to be used in the search for locally better solutions. The global information defines a set of necessary conditions which are used to avoid returning to any previously explored basis.

The problem to be treated is a special case of (1) where $k = 2$ and all the functions are linear. It can be stated as follows:

*P*6. Find values for the policy and behavioral variables $x \in R^{n^1}$ and $y \in R^{n^2}$ such that

$$f_1 = \max_{x \geqq 0} \{c_1 x + d_1 y\} \quad \text{where } y \text{ solves} \tag{6.1}$$

$$f_2 = \max_{y \geqq 0} \{c_2 x + d_2 y\} \tag{6.2}$$

$$\text{subject to} \qquad A^1 x + A^2 y \geqq b \tag{6.3}$$

where A^1 is $(m \times n^1)$ and A^2 is $(m \times n^2)$. Typical column vectors in A^1 and A^2 will be denoted by A_i^1 and A_j^2, and for convenience it is assumed that rank $(A^2) = m$.

Three related linear programming (LP) problems will now be defined. Each of these problems plays a role in the development of the algorithm.

*P*7. For any given (kth) set of nonnegative values for the policy variables $x = x^{(k)} \geqq 0$, find

values for the behavioral variables y, such that

$$f_2 = \max_{y \geq 0} \{c_2 x^{(k)} + d_2 y\} \tag{7.1}$$

subject to

$$A^2 y \geqq b - A^1 x^{(k)} \tag{7.2}$$

P7 will be referred to as the "behavioral" or "inner" LP problem; there is no guarantee that P7 will have a feasible solution for any given set of values $x^{(k)} \geqq 0$.

P8. Given a basis set B_k from A^2 that is optimal with respect to the behavioral LP problem P7, find values of $(x, y^{(k)})$ such that

$$f_1 = \max \{c_1 x + d_1 y^{(k)}\} \tag{8.1}$$

subject to

$$A^1 x + B_k y^{(k)} \geqq b \tag{8.2}$$

$$x, y^{(k)} \geqq 0. \tag{8.3}$$

The basis B_k is termed a *behavioral optimal basis* (BOB) if the associated reduced cost coefficients of the behavioral objective function are nonnegative, i.e., if

$$c_{2k} B_k^{-1} A_j^2 - c_{2j} \geqq 0, \quad \text{for all } j = 1, \ldots, n^2.$$

Note that behavioral optimality is unaffected by the settings for the policy variables x. The values for the policy variables only affect the feasibility of a behavioral basis.

P8 will be referred to as the "policy" or "outer" LP problem and is defined only for behavioral variables (activities) that are members of B_k.

P9. Final values of x and y such that

$$f_1 = \max \{c_1 x + d_1 y\} \tag{9.1}$$

subject to

$$A^1 x + A^2 y \geqq b \tag{9.2}$$

$$x, y \geqq 0 \tag{9.3}$$

Clearly the solution to P9 provides an upper bound for the solution to P6.

Solution to P6

By comparing problems P6 and P7, it can be seen that if P7 has an optimal feasible solution $y^{(k)}$, then $(x^{(k)}, y^{(k)})$ is a feasible solution to the bi-level linear program P6. If there are no feasible solutions to P7, then $x = x^{(k)}$ is not a feasible setting for the policy variables in P6.

In addition, if we are given a behavioral optimal basis B_k to the behavioral LP problem P7, then it can be shown that values for $(x^{(k)}, y^{(k)})$ satisfying the constraint set for the policy LP problem P8 (i.e. 8.2 and 8.3) are also feasible solutions to P6. Now let $x = x^*$ be an optimal feasible setting for the policy variables in P6. Then, from the definition of P6, there exists a basic optimal feasible solution to the behavioral LP problem P7:

$$y^* = B_*^{-1} b - B_*^{-1} A^1 x^* \geq 0 \text{ for } A_j^2 \in B_*$$

$$= 0 \qquad \text{otherwise,}$$

such that $f_1^* = c_1x^* + d_1y^*$ is the optimal value for the policy objective function and B_* is the (BOB) for P7. The relationship between the solutions for P6 and P8 are clarified in the following theorem.

Theorem 1

Given there exists an optimal feasible solution to the bi-level programming problem P6, there exists a (BOB), B_*, such that the corresponding basic optimal feasible solution to the policy LP problem P8 is an optimal feasible solution to P6.

As a corollary to Theorem 1 we have:

Corollary 1

An optimal feasible solution to the bi-level programming problem P6 can be represented as a basic feasible solution to problem P9.

Theorem 1 suggests an alternative to solving P6 directly. By concentrating on the solution to P8 as it varies with the choice of (BOB), the solution to P6 will eventually be uncovered. Candler and Townsley offer a systematic way of moving from one (BOB) to another without retracing any path already examined. By focusing on the reduced cost coefficients of the variables not in the (BOB), their algorithm provides a monotonic decrease in the number of behaviorally optimal bases which need to be examined.

3. STRUCTURAL CONSIDERATIONS

3.1 *Indifference points and nonexistence of solutions*

It was mentioned previously that unlike the general mathematical program, the multi-level program may not possess a solution even when f_1 and g^{m^i} are continuous over the compact sets X^i, $i = 1, \ldots, n$. To see this, let us consider problem (1) for the case where $n = 2$. Suppose that the outside player selects a point x^*. The inside player is then faced with a simple maximization problem parameterized by the vector x^*. In certain instances, the solution set to this problem may contain more than one member. For example, if all the constraint functions were linear, it is possible that $y^*(x^*)$, the set of all solutions to the inside player's problem for $x = x^*$ fixed, might consist of some nontrivial subset of a hyperplane. This would mean that the inside player would be indifferent to any point on that hyperplane; however, the outside player might not experience the same indifference with respect to his objective function. His best result might only be realized at one particular point in $y^*(x^*)$, but there may be no way to induce the inside player to show any preference for that point. It may further be true that if the outside player chooses any point other than x^*, his maximum payoff will never be realized. This situation is illustrated in the following example.

$$f_1 = \max_{x \geqq 0} \left\{ (x_1, x_2) \begin{pmatrix} -2 & -4 \\ -3 & -1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right\}$$

$$= \max_{x \geqq 0} \{ -(2y_1 + 3y_2)x_1 - (4y_1 + y_2)x_2 \} \tag{10.1}$$

where y solves

$$f_2 = \max_{y \geqq 0} \left\{ (x_1, x_2) \begin{pmatrix} 1 & 3 \\ 4 & 2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right\}$$

$$= \max_{y \geqq 0} \{ (x_1 + 3x_2)y_1 + (4x_1 + 2x_2)y_2 \} \tag{10.2}$$

subject to

$$x_1 + x_2 = 1, \quad y_1 + y_2 = 1. \tag{10.3}$$

Note, the structure of (10) is identical to that of a bimatrix game but now the order of play is sequential rather than simultaneous. The solution to the inside player's problem y as a function

of x is

$$y(x) = \begin{cases} (1,0) & \text{for } x_1 + 3x_2 > 4x_2 + 2x_1;\ \text{i.e., } x_1 < 1/4 \\ y_1 + y_2 = 1 & \text{for } x_1 = 1/4 \\ (0,1) & \text{for } x_1 > 1/4. \end{cases}$$

Substituting these values into (10.1) the outside player's problem becomes

$$f_1 = \max_{x \geqq 0} \begin{cases} -2x_1 - 4x_2 & ;\ x_1 < 1/4 \\ -2y_1 - 3/2 (0 \leqq y_1 \leqq 1); & x_1 = 1/4 \\ -3x_1 - x_2 & ;\ x_1 > 1/4 \end{cases} \tag{11}$$

subject to

$$x_1 + x_2 = 1.$$

At $x_1 = 1/4$, f_1 is not a well-defined function and an attempt to solve (11) leads to difficulties. This can be seen by substituting $1 - x_1$ for x_2 and plotting the value of the objective function f_1 as x_1 varies between zero and one. The results are shown in Fig. 1.

The largest payoff ($f_1 = -1.5$) for the outside player occurs when he selects $x = (1/4, 3/4)$ and the inside player selects $y = (0, 1)$. There is no guarantee, however, that the inside player will choose $(0, 1)$ since he is indifferent to any point on the line $y_1 + y_2 = 1$. The only way to assure this selection is for the outside player to pick a point such that $x_1 > 1/4$, say

$$x^* = (1/4 + \epsilon,\ 3/4 - \epsilon)$$

where $\epsilon > 0$ is arbitrarily small. The corresponding payoff is $f_1^* = 1.5 - 2\epsilon$ which is not the best result that he could have achieved. Thus, there is no sure way for the outside player to realize his maximum payoff.

This result suggests that some type of cooperation, perhaps in the form of side payments, would work to one advantage of both players. Such a change in the rules is outside the context of the present model and will not be considered here. Note that if cooperation among the players were permitted, it might be mutually beneficial for one player to accept a lower payoff than he might ordinarily receive in order for another player to receive a much larger payoff. The difference could then be split. In Section 4, we demonstrate the equivalence between the bi-level programming problem and the standard form of the mathematical program. In the accompanying reformulation, ambiguities or indifferences arising in solution values will be resolved by giving the outside player complete control over any multiple optimal solutions that the inside player may have. That is, while the original problem may not have a solution, we circumvent this difficulty by allowing the outside player more control than the original model permitted. In Section 5.4 we provide a check for existence.

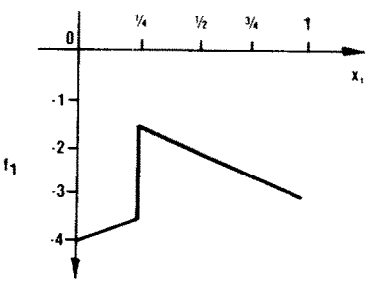

Fig. 1. Bi-level program with no solution.

3.2 *Significance of order of play*

Unlike the rules of noncooperative game theory where each player must choose a strategy simultaneously, the definition of multi-level programming requires that the outer player move first. In order to demonstrate the significance of the order of play, let us reverse the structure of problem (10). The new problem becomes

$$f_2 = \max_{y \geqq 0} \{(x_1 + 3x_2)y_1 + (4x_1 + 2x_2)y_2\} \tag{12.1}$$

where x solves

$$f_1 = \max_{x \geqq 0} \{-(2y_1 + 3y_2)x_1 - (4y_1 + y_2)x_2\} \tag{12.2}$$

subject to

$$x_1 + x_2 = 1 \tag{12.3}$$

$$y_1 + y_2 = 1. \tag{12.4}$$

The solution to the inside problem (12.2) and (12.3) for y fixed is

$$x(y) = \begin{cases} (1,0) & \text{for } -2y_1 - 3y_2 > -4y_1 - y_2; \text{ i.e., } y_1 > 1/2 \\ x_1 + x_2 = 1 & \text{for } y_1 = 1/2 \\ (0,1) & \text{for } y_1 < 1/2. \end{cases}$$

We can now rewrite the outside problem by substituting $x(y)$ into (12.1); i.e.,

$$f_2 = \max_{\substack{y \geqq 0 \\ y_1 + y_2 = 1}} \left\{ \begin{array}{ll} y_1 + 4y_2 & ; y_1 > 1/2 \\ (3 - 2x_1)y_1 + (2x_1 + 2)y_2 & ; y_1 = 1/2 \\ 3y_1 + 2y_2 & ; y_1 < 1/2 \end{array} \right\}$$

$$= \max_{0 \leqq y_1 \leqq 1} \left\{ \begin{array}{l} 4 - 3y_1; \; y_1 > 1/2 \\ 5/2 \quad ; y_1 = 1/2 \\ 2 + y_1; \; y_1 < 1/2. \end{array} \right\}$$

The solution $y^* = (1/2, 1/2)$ can readily be determined from the plot of f_2 for $0 \leqq y_1 \leqq 1$ given in Fig. 2.

The corresponding solution x^* for the inside player is any point within the set $\{x_1, x_2)$: $x_1 + x_2 = 1$, $x_1 \geqq 0$, $x_2 \geqq 0\}$. Thus, as in problem (10), the inside player is indifferent to a range of

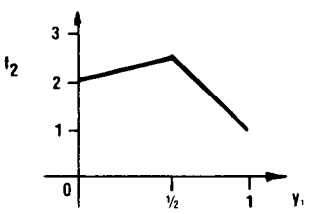

Fig. 2. Solution to problem 12.

Table 1. Significance of order of play

	Problem (10)	Problem (12)	Bimatrix Game
Solution (x)	(1/4, 3/4)	$x_1 + x_2 \quad 1$	(1/4, 3/4)
Payoff (f_1)	-3/2	-5/2	-5/2
Solution (y)	(0, 1)	(1/2, 1/2)	(1/2, 1/2)
Payoff (f_2)	5/2	5/2	5/2

points but now the outside player will receive the same payoff regardless of the inside player's choice.

Further, if we assume that the outside player in problem (10) achieves his maximum payoff, the two problems can be compared at their solution points. This comparison is presented in Table 1.

The last column in Table 1 represents the solution to the corresponding bimatrix game (see Bard[3]). It can be seen that if the player who controls the x variable is given the first move, he can realize a greater payoff than had he been assigned a different position. Of course, in most practical settings the order of play is determined logically by the underlying dynamics (e.g., by government regulation and industry's response) so reversing the order would make no sense.

4. REFORMULATION OF THE BI-LEVEL PROGRAMMING PROBLEM

In order to put problem (1) into a more manageable form, (when $n = 2$), we will make use of the Kuhn–Tucker conditions associated with the inside player's problem. To begin, let us restate the general bi-level programming problem for the case where no restrictions are placed on the attending functions; that is, find vectors $x \in R^{n^1}$ and $y \in R^{n^2}$ (where $n^1 + n^2 = \bar{n}$) to

$$\max_x f_1(x, y) \text{ where } y \text{ solves} \tag{13.1}$$

$$\max_y f_2(x, y: x) \tag{13.2}$$

subject to

$$g(x, y) \geqq 0 \tag{13.3}$$

where

$$f_1, f_2\colon R^{\bar{n}} \to R \text{ and } g\colon R^{\bar{n}} \to R^m.$$

The inside player's problem defined by (13.2) and (13.3) for x fixed is generally a nonconvex program and often difficult to solve since it may possess local optima. This means that even if the first and second order optimality conditions hold at a point, there is no guarantee that this point is a global, rather than local solution. If we now assume that f_2 and g are smooth, and that f_2 and g are concave in y for x fixed, then a necessary and sufficient condition for (y^*, u^*) to solve (13.2) and (13.3) is that the following first order conditions are satisfied.

$$\nabla_y f_2(x, y^*) + u^* \nabla_y g(x, y^*) = 0 \tag{14.1}$$

$$\langle u^*, g(x, y^*) \rangle = 0 \tag{14.2}$$

$$g(x, y^*) \geqq 0 \tag{14.3}$$

$$u^* \geqq 0 \tag{14.3}$$

where u is an m-dimensional row vector of Kuhn–Tucker multipliers and ∇ is the gradient operator. This leads to the following theorem.

Theorem 2

A necessary condition that (x^*, y^*, u^*) solves (13) is that (y^*, u^*) satisfy conditions (14.1)–(14.4) for $x = x^*$ fixed.

In light of Theorem 2, it is possible to reformat the bi-level program (13) as a standard mathematical program. The resulting problem is:

$$\max_{x,y,u} f_1(x, y) \tag{15.1}$$

subject to

$$\nabla_y f_2(x, y) + u\nabla_y g(x, y) = 0 \tag{15.2}$$

$$\langle u, g(x, y)\rangle = 0 \tag{15.3}$$

$$g(x, y) \geqq 0 \tag{15.4}$$

$$u \geqq 0 \tag{15.5}$$

Problem (15) is also a nonconvex program and, in general, no explicit solution technique exists that will reliably produce a global optimum. Additional restrictions must be placed on the functions if any progress is to be made towards obtaining a solution. Accordingly, consider the case where all the functions are linear. This leads to problem (6) which, when put into the form of (15), becomes

$$\max\{c_1x + d_1y\} \tag{16.1}$$

subject to

$$c_2 + uA^2 + u_2I_{n^2} = 0 \tag{16.2}$$

$$\langle u, (A^1x + A^2y - b)\rangle + \langle u_2, y\rangle = 0 \tag{16.3}$$

$$A^1x + A^2y \geqq b \tag{16.4}$$

$$x \geqq 0,\ y \geqq 0,\ u \geqq 0,\ u_2 \geqq 0 \tag{16.5}$$

where u^2 is an n-dimensional row vector of Kuhn–Tucker multipliers associated with the nonnegativity constraint of the inside problem, $y \geqq 0$, and I_{n^2} is an n_2-dimensional identity matrix. The only constraint in (16) that is nonlinear is (16.3), the complementary slackness condition, which has the general form

$$\sum_{i=1}^{m} u_i g_i = 0.$$

By observing that both u_i and g_i must be greater than or equal to zero at the solution we get the following lemma.

Lemma 1

The complementarity constraint (15.3) can be replaced by

$$\sum_{i=1}^{m} \min(u_i, g_i(x, y)) = 0 \tag{17}$$

without altering the solution of (15); that is, if (x^*, y^*, u^*) solves (15) it will also solve the new problem created by replacing (15.3) with (17).

Condition (17) as it is written is not yet in a useful form and, in fact, may seem more of a burden than a help because the functions are no longer smooth. Nevertheless we may rewrite (17) as follows:

$$\sum_{i=1}^{m} \{\min(0, (g_i - u_i)) + u_i\} = 0.$$

Making one more transformation in this sequence by replacing $g_i - u_i$ with a new set of variables w_i, $i = 1, \ldots, m$, we get

$$\sum_{i=1}^{n} \{\min(0, w_i) + u_i\} = 0 \tag{18.1}$$

$$w_i - g_i + u_i = 0, \quad i = 1, 2, \ldots, m \tag{18.2}$$

which for our purposes, is the equivalent of (15.3) that can be exploited with most advantage. The m components of the vector w will be referred to as the auxiliary variables. As can be seen, (18.1) is a piecewise linear function in w and u but more importantly, when taken together with (18.2), these two equalities provide a separably representation for (15.3) in terms of g and u.

Let us now return to problem (16) and replace (16.3) with the equivalent form given by (18). The following problem results.

$$\max c_1 x + d_1 y \tag{19.1}$$

subject to

$$c_2 + uA^2 + u_2 I_{n^2} = 0 \tag{19.2}$$

$$\sum_{i=1}^{m+n^2} \{\min(0, w_i) + u_i\} = 0 \tag{19.3}$$

$$w - A^1 x - A^2 y + b + u = 0 \tag{19.4}$$

$$w_2 - y + u_2 = 0 \tag{19.5}$$

$$A^1 x + A^2 y \geqq b \tag{19.6}$$

$$x \geqq 0, \; y \geqq 0, \; u \geqq 0, \; u_2 \geqq 0 \tag{19.7}$$

where w_2 is an n^2-dimensional vector of auxiliary variables.

Problem (19) is still a nonconvex program, but is now in a much more manageable form than either (6) or (16). Because all the functions are separable in the variables x, y, u and w we can use an existing algorithm (e.g. Falk[18] or Beale and Tomlin[5]) to obtain global solutions. In the next two sections we present some examples that demonstrate the computation aspects of this formulation.

5. COMPUTATIONAL IMPLICATIONS AND EXPERIENCE

A common requirement in the development and use of algorithms designed to solve general classes of optimization problems is that the functions be continuous and smooth, (e.g. see Cabot[11] or Fiacco and McCormick[22]). This requirement assures the existence of first order differential information which gives direction in the search for a Kuhn–Tucker point. In the reformation of the bi-level program (6) into a nonconvex program (16) we have sacrificed smoothness for separability; that is, all the functions can now be written in the following form:

$$f(x) = \sum_{j=1}^{n} f_j(x_j).$$

The traditional method for treating separable problems involves calculating piecewise linear approximations of the associated functions and applying a modification of the simplex method to the resulting problem (see, e.g. Miller[30]). The modification amounts to a restriction on the usual manner of selecting variables to enter and leave the basis and will yield a local but not necessarily a global solution to the approximate problem.

An algorithm for finding global solutions to nonconvex separable problems was developed by Falk and Soland[20]. The method is based on the branch and bound philosophy and yields a (generally infinite) sequence of points whose cluster points are global solutions of the problem. The implementation of the method is limited by the necessity of computing convex envelopes of the functions involved, although a number of applications have been shown possible when the functions exhibit special structure (e.g. concave or piecewise linear).

The inherent limitations that special problem structures impose have been overcome by the introduction of two algorithms independently developed by Beale and Tomlin[5] and Falk[18]. For this paper, we have used the programming code MOGG based on the algorithm proposed by Falk and written by Grotte[24].

The algorithm itself is based on branch and bound techniques and works by enclosing the feasible region of the separable nonconvex program in a linear polyhedron which is then divided into disjoint subsets. A lower bound on the optimal value of the problem is found by minimizing the objective function over each of these subsets and selecting the smallest value obtained. A check for the solution is made which, if successful, yields a global solution of the piecewise linear approximation to the separable nonconvex program. If the check fails, the subset corresponding to the smallest lower bound is further subdivided into either two or three new linear polyhedra and the process continues as before with new and sharper bounds being determined. The process is finite and terminates with a global solution of the approximate problem.

5.1 *Example* 1

For purposes of illustration, we will first examine the problem presented by Candler and Townsley to test their iterative scheme. This example, along with three others, will be used to highlight the computational aspects of the approach developed above.

The problem is to find vectors $x \in R^2$ and $y \in R^3$ that

$$\max_{x \geqq 0} \{8x_1 + 4x_2 - 4y_1 + 40y_2 - 4y_3\} \text{ where } y \text{ solves}$$

$$\max_{y \geqq 0} \{-x_1 - 2x_2 - y_1 - y_2 - 2y_3\}$$

subject to

$$\begin{aligned} y_1 - y_2 - y_3 &\geqq -1 \\ -2x_1 + y_1 - 2y_2 + 0.5y_3 &\geqq -1 \\ -2x_2 - 2y_1 + y_2 + 0.5y_3 &\geqq -1 \end{aligned}$$

Rewriting this problem in the form of (19) we get

$$\max \{8x_1 + 4x_2 - 4y_1 + 40y_2 + 4y_3\}$$

subject to

$$\sum_{i=1}^{6} \{\min(0, w_i) + u_i\}$$

$$\begin{aligned} -u_1 - u_2 - 2u_3 - u_4 &= -1 \\ u_1 + 2u_2 - u_3 - u_5 &= -1 \\ u_1 - 0.5u_2 - 0.5u_3 - u_6 &= -2 \end{aligned}$$

$$w_1 - y_1 + y_2 + y_3 + u_1 = 1$$

$$w_2 + 2x_1 - y_1 + 2y_2 - 0.5y_3 + u_2 = 1$$

$$
\begin{array}{llllll}
w_3 & +2x_2+2y_1-y_2-0.5y_3 & +u_3 & & & =1 \\
w_4 & -y_1 & & +u_4 & & =0 \\
w_5 & -y_2 & & & +u_5 & =0 \\
w_6 & -y_3 & & & +u_6 & =0
\end{array}
$$

$$y_1 - y_2 - y_3 \geqq -1$$

$$-2x_1 + y_1 - 2y_2 + 0.5y_3 \geqq -1$$

$$-2x_2 - 2y_1 + y_2 + 0.5y_3 \geqq -1$$

$$x \geqq 0,\ y \geqq 0,\ u \geqq 0,\ w \geqq 0$$

Ordinarily, the algorithm (MOGG) gives only an approximate answer to the nonconvex program because the original functions are replaced with their piecewise linear approximations. A related problem is then set up and solved. In the case of (19) though, no replacements will be made because all the functions are already piecewise linear. Therefore, MOGG will produce an exact solution to problem (19) and hence (6).

The computations generated by a branch and bound algorithm are customarily depicted by a branch and bound tree. In this type of arrangement the nodes of the tree correspond to the related linear subproblems, while the branches of the tree correspond to the set on which the branching variables are defined. In other words, the feasible region associated with any node in the tree is, in part, determined by the hyperrectangle defined by the branches connecting that node with the origin. With each extension of the tree, the approximation to the feasible region becomes increasingly sharper until no more branching is possible. A solution is obtained at the node where the best upper bound and the best lower bound converge. For problems where all the nonlinear functions are piecewise linear and contain only one break, the tree will be at most m branches deep, where m is the number of nonlinear functions. This means that the domain of each of the associated nonlinear variables can be divided into at most two segments so a maximum of $2^{m+1}-1$ subproblems might have to be solved.

When branch and bound techniques are used the solution is often uncovered before it is recognized. Here, the solution was uncovered at the 14th stage after 28 subproblems had been solved but was not recognized until the 51st stage. In all, 103 subproblems had to be solved. The branch and bound tree for this problem will not be presented because of its extensive length. The solution values are:

$$x^* = (0.0, 0.9) \qquad u^* = (0.0, 1.0, 3.0, 6.0, 0.0, 0.0)$$

$$y^* = (0.0, 0.6, 0.4) \qquad w^* = (0.0, -1.0, -3.0, -6.0, 0.6, 0.4)$$

$$f_1^* = 29.2 \qquad f_2^* = -1.9.$$

In this example, the number of nonlinear (auxiliary) variables is six. The theoretical upper limit on the number of subproblems that might have to be solved in therefore, 127. As can be seen, the number of subproblems actually solved was undesirably close to the upper limit. If this were true in general, there might be reason to call into question the efficiency (but not necessarily the usefulness) of the approach. Subsequent examples prove otherwise.

5.2 *Example* 2

$$\max_{x \geqq 0} \{2x_1 - x_2 - 0.5y_1\} \quad \text{where } y \text{ solves}$$

$$\max_{y \geqq 0} \{-x_1 - x_2 + 4y_1 - y_2\}$$

subject to

$$2x_1 \quad -y_1 + y_2 \geqq 2.5$$

$$-x_1 + 3x_2 \quad -y_2 \geqq -2$$

$$-x_1 - x_2 \qquad \geqq -2$$

When this problem is recast as a standard mathematical program, two new sets of variables comprising auxiliaries and Kuhn–Tucker multipliers, respectively, are created. Each set is pairwise associated with the four inequalities containing the behavioral variable y.

In theory, a total of 31 subproblems might have to be solved before convergence takes place. In fact, the algorithm converged in the seventh stage after 15 subproblems had been solved. The branch and bound tree is shown in Fig. 3. The two numbers adjacent to each node represent the best upper and lower bounds for that subproblem. A bar in the place of the best upper bound indicates that no corresponding feasible point to the approximate problem exists. The numbers along the branches refer to the branching (auxiliary) variables associated with the preceding node. The left branch (+) indicates that the particular variable was permitted to range over the set of positive real numbers while the right branch (−) indicates the same for the set of negative real numbers. The bars appearing below the nodes indicate that either the lower bounds of the associated subproblems are all greater than the current best upper bound or that they are infeasible and, therefore, cannot contain the solution.

The solution values for the decision and auxiliary variables are as follows:

$$x^* = (1, 0) \qquad u^* = (4, 1.5, 0, 0)$$
$$y^* = (0.5, 1) \qquad w^* = (-4, -1.5, 0, 5, 1).$$

The optimal value of the outside player's objective function $f_1^* = 1.75$, while the corresponding payoff for the inside player $f_2^* = 0$.

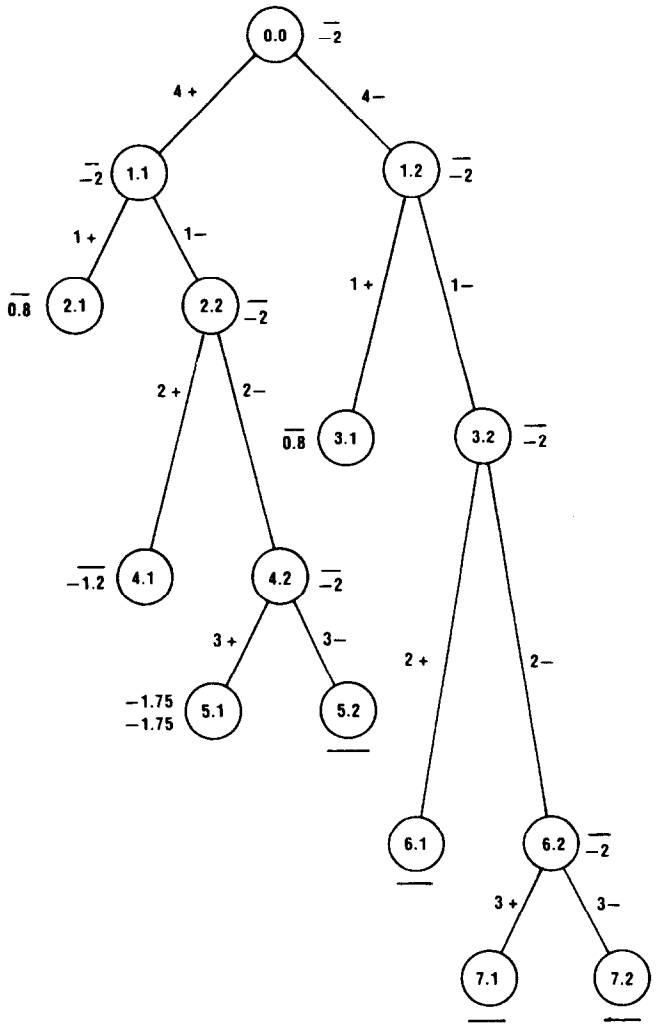

Fig. 3. Branch and bound tree for example 2.

5.3 *Sensitivity of MOGG to the size of the feasible region*

At each iteration MOGG solves a linear program whose objective function value at the solution provides a lower bound on the associated subproblem. If the solution vector to the linear program is feasible to the original problem (in its approximate form), then it also produces an upper bound to this problem. The availability of an upper bound in the early stages of the algorithm usually speeds convergence because a comparison can then be made with current lower bounds. No branching will take place from a node whose lower bound is greater than the best upper bound in the tree. In Example 2, a feasible point was not found until the 10th subproblem was solved; while in Example 1, 82 subproblems had to be solved before an upper bound was produced. In both cases, this point happened to be the solution as well as the only feasible point uncovered.

For a given problem, there does not seem to be much we can do, short of redesigning the algorithms to produce a feasible point at an earlier stage than was otherwise noted. Nevertheless, it may be instructive to know the way in which MOGG's rate of convergence varies as the parameters in the original problem are perturbed. To get an understanding of this relationship, we investigated the convergence properties of Example 2 for a range of r.h.s. values. Specifically, the value of b_1, the resource parameter associated with the first inequality was varied between 1.5 and 3.5 in increments of 0.1. This had the effect of first loosening and then gradually shrinking the feasible region of the original problem. For $b_1 < 3$ no change resulted in the rate of convergence. Fifteen subproblems were set up and solved in all instances. For $b_1 \geqq 3$, however, MOGG converged on the first iteration, i.e. the solution vector for the first subproblem was feasible to the original problem and the respective objective function values were equal. Similar behavior was observed as b_2 was varied between -2.5 and -1.0. In this case, the threshold value for immediate convergence was $b_2 = -1.5$.

These results tentatively indicate that the size of the feasible region plays an important role in the efficiency of the algorithm. They also lend support to the proposition that the earlier an upper bound is determined, the faster will be the rate of convergence.

5.4 *Determining when no solution exists*

In Section 3.1, it was shown that the multi-level programming problem may not possess a solution when a particular player is indifferent between a number of points. In the reformulated version (15) of the bi-level program (13), the potential for uncertainty in the existence of a solution is eliminated because a single player now controls all the variables. Nevertheless, it is important to know whether or not the values computed for this problem would actually be realized in the play of the game. If the inside player is faced with multiple solutions, there is no guarantee that he will select the value that optimizes the outside player's problem.

In order to determine if the current solution (x^*, y^*) actually solves the bi-level program or whether no solution exists at all, we must examine the inside player's problem for x fixed at x^*. Let $y(x^*)$ be the corresponding solution set. If every value in $y(x^*)$ produces a unique value for the outside player's objective function f_1, then (x^*, y^*) is indeed the solution. Unfortunately, the explicit form of $y(x^*)$ is rarely known, but even if it were, it would be a tedious, if not impossible matter to determine f_1 for each member. As a more practical check, we propose solving the following optimization problem:

$$\bar{f}_1 = \min_y f_1(x^*, y) \tag{20.1}$$

subject to

$$g(x^*, y) \geqq 0 \tag{20.2}$$

$$f_2(x^*, y) = f_2(x^*, y^*). \tag{20.3}$$

Constraint (20.3) determines $y(x^*)$ implicitly by assuring that the inside player realizes his maximum payoff. If the optimal value of the objective function $\bar{f}_1$ in (20.1) is equal to $f_1(x^*, y^*)$ then the current solution is achievable. Conversely, if $\bar{f}_1$ is less than $f_1(x^*, y^*)$ it becomes

immediately clear that the inside player has multiple optima, but more importantly, no solution exists for the original bi-level program.

For purposes of illustration, let us return to problem (10) and check to see if the solution that would have been derived from solving (13) is guaranteed to exist. Putting (10) into the form of (20) when $x^* = (1/4, 3/4)$ we get

$$f_1 = \min_{y \geqq 0} \left\{ -\frac{7}{2}y_1 - \frac{3}{2}y_2 \right\}$$

subject to

$$y_1 + y_2 = 1.$$

The solution to this problem occurs at $y = (1, 0)$; the corresponding value of the objective function $\bar{f}_1 = -7/2$. From Table 1 we see that $f_1^* = -3/2$ which is greater than $\bar{f}_1$. Therefore, our initial conclusion that problem (10) does not have a solution is confirmed.

6. THE MAX-MIN PROBLEM

The max–min problem is traditionally defined (Danskin[13]) as a two-stage optimization problem in which the minimizing (inside) player acts after the maximizing (outside) player has chosen his strategy. It can be viewed as a noncooperative, zero-sum game with perfect information taking the following form:

$$\max_{x \in X} \min_{y \in Y} f(x, y) \tag{21}$$

where x is an n^1-dimensional vector, y is an n^2-dimensional vector, and X and Y are assumed to be compact subsets of R^{n^1} and R^{n^2}. Problem (21) is a special case of the bi-level program (13) and therefore holds a special interest for us. To see the similarity between these two problems, let us rewrite (21) in the form of (13) by setting $f_1 = -f_2$. As a result, we get

$$\max_{x \in X} f(x, y)$$

where y solves

$$\max_{y \in Y} -f(x, y).$$

The original constraint region (13.3) is now disjoint and given by X and Y.

Danskin[13] has developed a basic theory of max–min which is analogous to the elementary theories of mathematical programming. In order to highlight equivalent concepts, let us define

$$\emptyset(x) = \min_{y} f(x, y) \tag{22}$$

and assume that $f(x, y)$ and its partial derivatives $f_{x_i}(x, y)$ with respect to x_i, $i = 1, \ldots, n^1$, are continuous. The problem then is to maximize $\emptyset(x)$, where $\emptyset(x)$ is a continuous function of x (since f is continuous on X).

The major difficulty in studying $\emptyset(x)$ is that this function, however smooth the original function $f(x, y)$, is not in general differentiable in the elementary sense. In particular, it may not be differentiable in x at the point yielding the maximum. Characteristically, $\emptyset(x)$ has sharp ridges; nevertheless, Danskin has shown that this function has a directional derivative D that can be explicitly computed for all directions. More formally, if $(v_1, \ldots, v_{n^1})$ represents a direction of unit length in the n^1-dimensional space, then

$$D_v \emptyset(x) = \min_{y \in Y_x} \sum_i v_i f_i(x, y),$$

the minimum being taken over the set Y_x of all those y's which yield the minimum against x in (22). Corners and sharp ridges in $\emptyset(x)$ occur only when Y_x consists of more than a single point.

In addition, he has shown that the Lagrange multiplier principle associated with the maximization problem for $\emptyset(x)$ is valid when the constraint set X is polyhedral. That is, if x^* maximizes $\emptyset(x)$ subject to p side conditions

$$\sum_i a_{ij}x_j \geqq b_j \qquad j = 1, \ldots, p,$$

then there exist nonegative $\lambda_1, \ldots, \lambda_p$ such that

$$D_v\psi(x^*) \leqq 0$$

for any direction v, where

$$\psi(x) = \emptyset(x) + \sum_{i,j} \lambda_j a_{ij}x_i.$$

Bram[10] has extended this result for the more general constraint set given by $X = \{x: g(x) \geqq 0\}$, where $g: R^n \to R^p$ is continuously differentiable for all x in X and $\nabla g_j(x) \neq 0$, $j = 1, \ldots, p$, everywhere. More recently, Schmitendorf[33] has developed necessary and sufficient conditions for the max–min problem while obtaining a Lagrange multiplier rule in the form of an equality rather than in inequality.

Thus, the framework exists for solving the max–min problem as it is traditionally formulated, although most such problems can only be solved implicitly. If the problem is now viewed as a specialized version of the bi-level program, however, we can recast it into a more manageable form and obtain solutions explicitly. This reformulation has the additional benefit of permiting us to extend the framework described above to the more general case where the players' strategy sets are jointly dependent.

Before applying the methodology of section 4 to the max–min problem, we will briefly discuss a few other solution techniques that are available for solving the strictly linear case.

6.1 *The general linear max–min problem*

In the problems that we have been considering, the decision of the maximizing player ordinarily influences both the objective function and the constraint region of the minimizing player. When all the functions are linear, (21) can be rewritten as follows to take this interdependence into account

$$\max_x \min_y \{cx + dy: A^1x + A^2y \geqq b, \quad x \geqq 0, \quad y \geqq 0\} \tag{23}$$

where $x, c \in R^{n^1}$, $y, d \in R^{n^2}$, A^1 is an $m \times n^1$ matrix, A^2 is an $m \times n^2$ matrix, and $b \in R^m$.

A number of applications which exhibit this structure can be found in Konno[25]. In addition, Falk[19] has cited a potential example which is directed at finding approximate, global maximizing points of convex functions defined over linear polyhedra. An algorithm which can be used to solve problems equivalent to (23) has been developed by Konno[25]. The algorithm yields an "ϵ-optimal solution" in a finite number of steps and is based on Ritter's cutting plane method.

Falk has also developed an algorithm based on branch and bound techniques by first demonstrating the equivalence of the two-stage max–min problem and the nonconvex program. His work is grounded in the following theorems, presented here to further characterize the properties of (23).

Theorem 3 (*Falk*[19])

Problem (23) is equivalent to a nonconvex program whose objective function is a piecewise linear, convex function and whose constraints describe the linear polyhedron $P(X)$, where

$P(X) = \{x \geqq 0: \exists y \geqq 0$ for which $A^1x + A^2y \geqq b\}$. If the feasible region for (22) is defined by

$$S = \{(x, y) \geqq 0: \quad A^1x + A^2y \geqq b\}$$

then $P(X)$ represents the projection of S onto R^{n^1}.

It will be said that a point (x^*, y^*) is an optimal solution of (23) if (a) y^* is optimal to the inside minimization problem for $x = x^*$ fixed; and (b) for all $x \in P(X)$,

$$cx^* + dy^* \geqq \min_y \{cx + dy: A^2y \geq b - A^1x, \ y \geqq 0\}.$$

Theorem 4. (*Falk*[19])

There exists a solution (x^*, y^*) of problem (23). Moreover, there is a solution such that x^* is a vertex of $P(x)$.

Theorem 5. (*Falk*[19])

There is a solution (x^*, y^*) of problem (23) which is a vertex of S.

Unfortunately, the results of Theorems 3–5 cannot be extended to the bi-level programming problem (6). The proof of Theorem 3 is based on the fact that both players are optimizing the same objective function, albeit in different directions. The best that can be done is to show that (6) is equivalent to maximizing a piecewise linear nonconvex function over a linear polyhedron. Such problems do not necessarily have vertex solutions. Further, we have already shown in Section 3.1 that (6) need not have a solution; however, it should be noted that the outside problem is bounded above on $P(x)$ by $\max\{c_1x + d_1y: (x, y) \in S\}$, and that the inside problem is bounded above on $P(y)$ (the projection of S onto R^{n^2}) by $\max\{c_2x + d_2y: (x, y) \in S\}$.

From Theorem 3 it can be seen that (23) is equivalent to a problem of maximizing a convex function over a linear polyhedron. Although a number of methods have been proposed to solve such problems (e.g. see Cabot[11] or Tui[35]), none can be directly applied here since each requires an explicit representation of the objective function and the constraint region. Neither $\emptyset(x)$ nor $P(x)$ meet this requirement.

By Theorem 5, we know that a solution of problem (23) may be found at a vertex of S. Hence, one possible way to find such a point would be to generate all vertices of $P(x)$ (e.g. see [2]) and test each one as a possible local solution by minimizing in y for fixed x. Rather than enumerate all vertices of S explicitly, Falk has developed an algorithm which will implicitly search all possible solutions and select the best of these as the global solution. The simplex method is used to obtain the upper bounds required in the branch and bound algorithm and to select candidate solutions. Branching takes place on individual variables and is effected by holding variables out of the basis.

Finally, Gallo and Ülkücü[23] address the same max–min problem as Falk but take a completely different approach. Whereas Falk shows that this problem is equivalent to a concave minimization problem, Gallo and Ülkücü reformulate it as a nonconvex program with a linear objective function and a nonconvex feasible region. They give necessary and sufficient optimality conditions for the equivalent problem and use these conditions to develop an explicit search routine. The proposed algorithm is of the cutting plane type, generating a sequence of enlarging polyhedra in a manner similar to that taken by Tui[35] for solving concave minimization problems. Each iteration of the algorithm involves solving $n + 1$ linear programs, where n is the number of neighboring vertices associated with any nondegenerate vertex. However, because n of the programs differ from each other by only one column, it is possible to significantly reduce the overall computational requirements by linking the solutions. No proof of finite convergence is given but the authors note that cycling has not been a problem.

6.2 *Sample computations*

Two sample problems will be presented in this section in order to demonstrate the computational aspects of solving the linear max–min problem by the method developed in Section 4. The first example was cited by Falk[19] who used a branch and bound algorithm based on linear programming techniques to obtain a solution. Candler and Townsley also

applied their iterative scheme to this problem. The second example has not been studied elsewhere, but was constructed with the specific (but not germane) motive of ensuring that the global solution occurred at a particular vertex of $P(x)$. Two other local solutions are known to exist, each occurring at different vertices.

Example 3

$$f = \max_{x \geqq 0} \min_{y \geqq 0} \{2x_1 - x_2 - 8y\}$$

subject to

$$\begin{aligned} x_1 + x_2 + y &\leqq 3 \\ x_1 \quad\quad - y &\leqq 0 \\ -x_1 - x_2 + y &\leqq 1 \\ x_1 + x_2 \quad &\leqq 2. \end{aligned}$$

Rewriting this problem in the form of (19) we get

$$\max_{x, y, u, w} \{2x_1 - x_2 - 8y\}$$

subject to

$$\sum_{i=1}^{4} \{\min(0, w_i) + u_i\} = 0$$

$$\begin{aligned} & & & & & & u_1 - u_2 + u_3 - u_4 &= 8 \\ w_1 & & + x_1 + x_2 + y + u_1 & & & & &= 3 \\ & w_2 & + x_1 \quad - y & & + u_2 & & &= 0 \\ & & w_3 \quad - x_1 - x_2 + y & & & + u_3 & &= 1 \\ & & & w_4 & \quad - y & & + u_4 &= 0 \\ & & x_1 + x_2 + y & & & & &\leqq 3 \\ & & x_1 \quad - y & & & & &\leqq 0 \\ & & - x_1 - x_2 + y & & & & &\leqq 1 \\ & & x_1 + x_2 & & & & &\leqq 2 \\ & & & & & & x \geqq 0, \ y \geqq 0, \ u &\geqq 0. \end{aligned}$$

The solution occurred on the first iteration of MOGG so no branch and bound tree developed. The computed values are

$$\begin{aligned} f^* &= -7 \\ x^* &= (1, 1) \quad u^* = (8, 0, 0, 0) \\ y^* &= (1) \quad\quad w^* = (-8, 0, 2, 1). \end{aligned}$$

Note that the first and second constraints are binding for the inside player's problem, but strict complementary slackness holds only for the first. In addition, observe that the maximum value of the objective function occurs at $(0, 0, 0)$ with $f = 0$, but the point $y = 0$ is not optimal with respect to the point $x = (0, 0)$. In fact $x = (0, 0)$, $y = (1)$ is a local solution with $f = -8$.

Finally, a perturbation analysis was performed on this example to test the sensitivity of MOGG's convergence rate to the size of the feasible region. The r.h.s. parameters of the original constraints were increased both separately and in unison in 0.1 increments. In each case, when a certain threshold value was reached, the number of subproblems that had to be solved jumped from one to either 15 or 17. This behavior tends to corroborate the findings presented in Section 5.3 implying a relationship between the relative size of the feasible region and the convergence properties of the algorithm.

Example 4

$$\max_{x \geqq 0} \min_{y \geqq 0} \{-x_1 + 0.1y_2\}$$

subject to

$$\begin{aligned} x_2 + y_1 - y_2 &\leqq 4 \\ x_1 - 2x_2 + 2y_1 + 2y_2 &\leqq 8 \\ 11y_1 + 2y_2 &\leqq 44 \\ -2x_1 + 2x_2 &\leqq 1 \\ x_1 + 4x_2 &\leqq 13 \\ x_1 - 1.5x_2 &\leqq 2. \end{aligned}$$

In this example, $P(x)$, the project of the feasible region S onto the x space is given explicitly by the intersection of the last three inequalities and the nonnegativity condition $x \geqq 0$. Because the resulting polyhedron is in two-dimensional space, it would be an easy matter to enumerate all the vertices of $P(x)$ and, by appealing to Theorem 4, solve the inside problem for each vertex and then pick out the solution. As was pointed out, however, enumerating vertices is often a difficult task even for polyhedra with a limited number of faces. Moreover, the work required to determine $P(x)$ when it is not given explicitly is usually comparable in scope to that of solving the original problem.

When Example 4 is recast as a nonconvex program, five auxiliary variables appear in the reformulation. The solution occurred and was recognized in the 22nd stage of MOGG after 45 subproblems had been solved. The associated branch and bound tree is depicted in Fig. 4. At the terminal node containing the solution the tree is five branches deep. The results are given below.

$$x^* = (5, 2) \qquad u^* = (0.5, 0.25, 0, 0, 0)$$

$$y^* = (2.75, 0.75) \qquad w^* = (-0.5, -0.25, 12.25, 2.75, 0.75).$$

The vector of Kuhn–Tucker multipliers indicates the first and second constraints are binding. The optimal value of the objective function $f^* = -2.55$. In addition, we note that this problem contains two other local solutions occurring at

$$(x, y) = \left((1, 3), \left(\frac{46}{13}, \frac{33}{13}\right)\right) \text{ and } ((2, 0), (3, 0)).$$

Only the second was uncovered by the algorithm and this was in the twenty-first stage.

7. CONCLUSIONS

The multi-level programming problem can be tentatively viewed as a standard mathematical program whose constraint region has been modified to include a series of implicitly defined functions. Although other problems such as those arising in the traditional max–min context may be classified in a similar manner, multi-level programming is now emerging in its own right

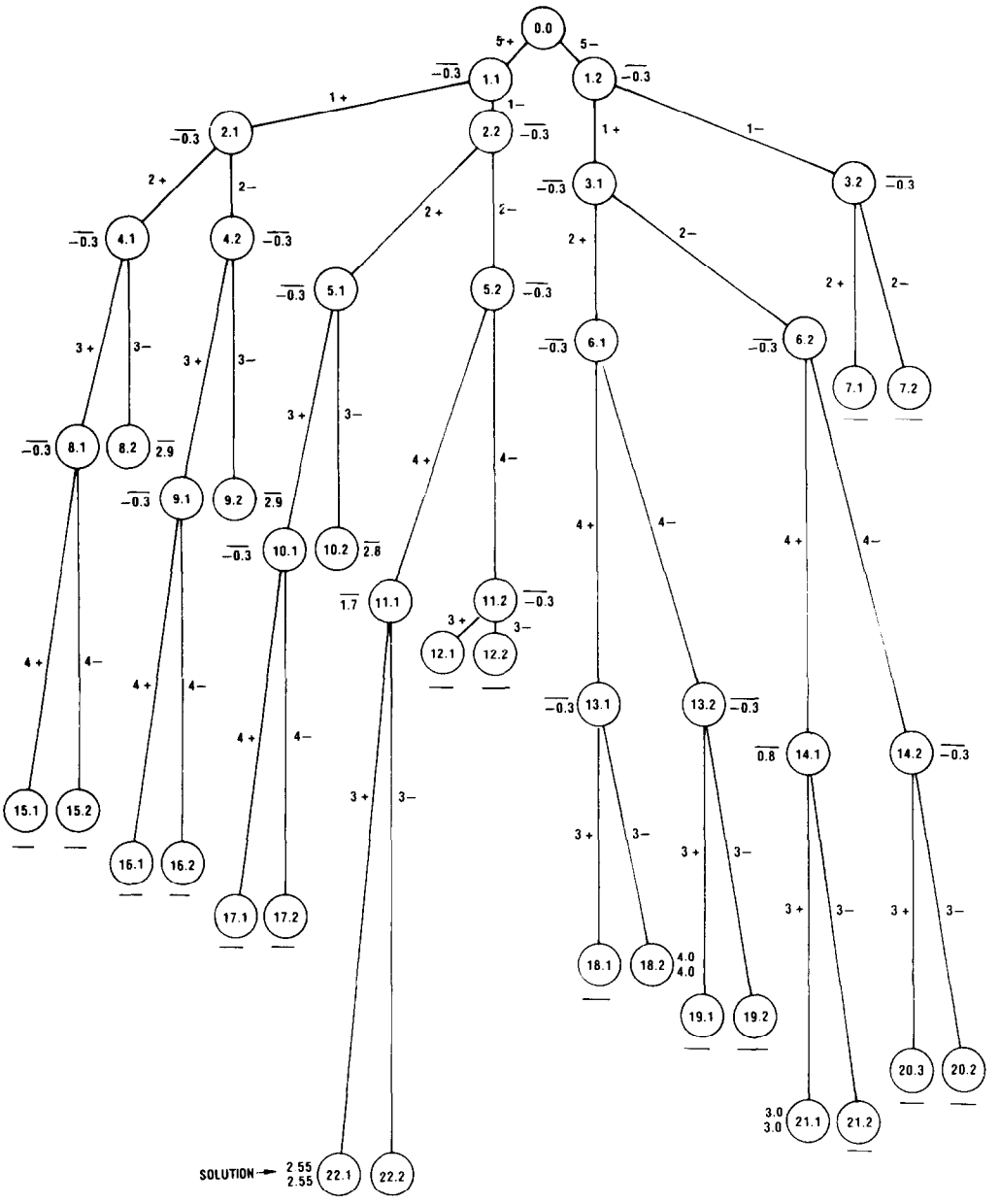

Fig. 4. Branch and tree for example 4.

as an independent component of optimization theory. Applications have been cited in such areas as government regulation and armed conflict, but potentially exist in all settings where opposing forces operate in a strictly competitive and open manner. Strong structural similarities can also be found within the theory of n-person, noncooperative games, the primary differences being limited to the order of play and the definition of strategy sets. In multi-level programming, the players no longer move concurrently but in a preassigned order, and must select their strategies from dependent rather than disjoint sets.

In this paper, the treatment of the multi-level program has been limited to the bi-level case. It was shown that the resulting program could be recast as a standard optimization problem characterized by an inherently nonconvex constraint region. Because one player rather than two now chooses the values of all the variables, a solution is guaranteed to exist.

In the reformulated program, the inside player is eliminated and the Kuhn–Tucker conditions associated with his problem are appended to the constraint region of the remaining player. When all the functions are linear, the resulting problem can be put into a separable form by modifying the complementary slackness condition associated with the inside player's problem. From a practical point of view, function separability is important because it enables us

to use an existing optimization routine to obtain global solutions. No algorithm currently available will reliably produce global solutions to the general nonconvex program.

If the original functions are not separable, it is often possible to reformulate them by adding a sequence of auxiliary variables to the constraint set; however, the algorithmic efficiency decreases as dimensionality increases. The advantage of the approach developed herein owes to the fact that separability has been achieved at very little cost in dimensionality. Further, when all the functions are linear, the first order stationarity condition for the inside player naturally leads to a set of linear equalities in the Kuhn–Tucker multipliers. This suggests that the use of the MOGG could readily be extended to problems where f_1 and f_2 are separable in x and y, and f_2 is concave in y for x fixed. Thus, we are now able to explicitly solve a general class of bi-level programs.

For all such programs, the algorithm is guaranteed to converge in a finite number of iterations. The few cases investigated, however, have shown that the rate of convergence is likely to depend upon the relative size of the constraint region. For the smaller region convergence appears to be immediate. As the region is enlarged a threshold is reached. At this point the number of subproblems that must be solved jumps significantly from one to between 50 and 70 percent of the theoretical upper limit. This is due in part to the fact that as the feasible region becomes less restricted the probability of the best lower bound coinciding with the actual solution decreases. For the bi-level program, this suggests that it might be possible to divide the constraint region in a more efficient manner than is currently prescribed.

REFERENCES

1. R. L. Armacost and A. V. Fiacco, Computational experience in sensitivity analysis for nonlinear programming. *Math. Prog.* **6**(3), 301–326 (1974).
2. M. Balinski, An algorithm for finding all vertices of a convex polyhedral set. *SIAM J.* **9**(1), 72–88 (1961).
3. J. F. Bard, *The application of nonconvex programming techniques to the equilibrium and the multi-level programming problems*, D.Sc. Dissertation, Department of Operations Research, The George Washington University (1979).
4. J. F. Bard and J. E. Falk, Computing equilibrium via nonconvex programming, *Technical Paper Serial T*-386, The George Washington University (1978).
5. E. M. L. Beale and J. A. Tomlin, Special facilities in a general mathematical programming system for nonconvex problems using ordered sets of variables. *Proceedings of the Fifth International Conference on Operations Research* (Edited by J. Lawrence), pp. 447–454, Tavistock Publications, London (1970).
6. J. W. Blankenship and J. E. Falk, Infinitely constrained optimization problems. *J. Optim. Theory Applics* **19**(2), 261–281 (1976).
7. J. Bracken and J. T. McGill, Computer program for solving mathematical programs with nonlinear programs in the constraints, *P*-801, Institute for Defense Analyses. Arlington, Virginia (1972).
8. J. Bracken and J. T. McGill, Mathematical programs with optimization problems in the constraints. *Ops Res.* **21**(1), 37–44 (1973).
9. J. Bracken, J. E. Falk and F. A. Miercort, A strategic weapons exchange allocation model. Ops Res. **25**, 968–976 (1977).
10. J. Bram, The Lagrange multiplier theorem for max–min with several constraints. *J. SIAM Appl. Math.* **14**(4), 665–667 (1966).
11. A. V. Cabot, *Variations on a cutting plane method for solving concave minimization problems with linear constraints.* Indiana University (1972).
12. W. Candler and R. J. Townsley, A linear multi-level programming problem (Unpublished paper) (1978).
13. J. W. Danskin, The theory of max–min with applications. *J. SIAM Appl. Math.* **14**(4), 641–664 (1966).
14. M. Davis and M. Maschler, The kernel of a comparative game. *Naval Res. Logist. Quart.* **12**, 223–259 (1965).
15. A. de Silva, *The application of formulas for nonlinear factorable programming to the solution of implicitly defined optimization problems*, D.SC. Dissertation, The George Washington University (1978).
16. B. C. Eaves, The linear complementarity problem. *Mgnt Sci.* **17**, 612–634 (1971).
17. J. F. Evans and F. J. Gould, Stability in nonlinear programming. *Ops Res.* **18**, 107–118 (1970).
18. J. E. Falk, An algorithm for locating approximate global solutions of nonconvex, separable problems, *Technical Paper Serial T*-262, The George Washington University (1972).
19. J. E. Falk, A linear max–min problem. *Math. Prog.* **5**, 169–188 (1973).
20. J. E. Falk and R. M. Soland, An algorithm for separable nonconvex programming problems. *Mgmt Sci.* **15**, 550–569 (1969).
21. A. V. Fiacco, Sensitivity analysis for nonlinear programming using penalty methods. *Math. Prog.* **10**(3), 287–311 (1976).
22. A. V. Fiacco and G. P. McCormick, *Nonlinear programming: Sequential Unconstrained Minimization Techniques.* Wiley, New York (1968).
23. G. Gallo and A. Ülkücü, Bilinear programming: An exact algorithm. *Math. Prog.* **12**, 173–194 (1977).
24. J. H. Grotte, Program MOGG—A code for solving separable nonconvex optimization problems *P*-1318, The Institute for Defense Analysis. Arlington, Virginia (1976).
25. J. H. Konno, Bilinear programming: Part II. Applications of bilinear programming, *Technical Report No.* 71-10, Operations Research House. Stanford, California (1971).
26. C. E. Lemke, Bimatrix equilibrium points and mathematical programming. *Mgmt Sci.* **11**, 681–89 (1965).
27. C. E. Lemke and J. T. Howson Jr., Equilibrium points of bimatrix games. *J. SIAM* **12**(2), 413–423 (1964).
28. R. D. Luce and H. Raiffa, *Games and Decisions.* Wiley, New York (1957).

29. O. L. Mangasarian, Characterization of linear complementarity problems as linear programs, *Computer Sciences Technical Report No.* 271, Computer Science Department, University of Wisconsin–Madison (1976).
30. C. E. Miller, The simplex method for local separable programming. *Recent Advances in Mathematical Programming* (Edited by R. L. Graves and P. Wolfe), pp. 89–100. McGraw Hill, New York (1963).
31. J. Nash, Non-cooperative games. *Ann. Math.* **54**, 286–295 (1951).
32. G. Owen, *Game Theory.* Saunders, Philadelphia (1968).
33. W. E. Schmitendorf, Necessary conditions and sufficient conditions for static min–max problems. *J. Math. Anal. and Appl.* **57**(3), 683–693 (1977).
34. A. W. Tucker, Solving a matrix game by linear programming. *IBM J. Res. Develop.* **4**(5) 507–517.
35. H. Tui, Concave programming under linear constraints. *Soviet Math. Dokl.* **4**, 1437–1440 (1964).