ELSEVIER

European Journal of Operational Research 88 (1996) 563–571

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

A simple Tabu Search method to solve the mixed-integer linear bilevel programming problem

U.P. Wen *, A.D. Huang

Department of Industrial Engineering, National Tsing Hua University, Kuang Fu Road, Hsinchu, Taiwan, R.O.C

Received November 1993; revised March 1994

Abstract

Multilevel programming is characterized as mathematical programming to solve decentralized planning problems. The models partition control over decision variables among ordered levels within a hierarchical planning structure of which the linear bilevel form is a special case of a multilevel programming problem. In a system with such a hierarchical structure, the high-level decision making situations generally require inclusion of zero-one variables representing 'yes-no' decisions. We provide a mixed-integer linear bilevel programming formulation in which zero-one decision variables are controlled by a high-level decision maker and real-value decision variables are controlled by a low-level decision maker. An algorithm based on the short term memory component of Tabu Search, called Simple Tabu Search, is developed to solve the problem, and two supplementary procedures are proposed that provide variations of the algorithm. Computational results disclose that our approach is effective in terms of both solution quality and efficiency.

Keywords: Mixed integer linear bilevel programming; Tabu Search; Heuristic algorithm

1. Introduction

Decentralized planning has long been recognized as an important decision making problem. Planners have commonly been arranged within a hierarchical administrative structure, each with independent and perhaps conflicting objectives. A multilevel programming problem is a model of a planner at one level of a hierarchy, having his objective function and decision space partially determined by other levels. Each planner's control instruments may allow him to influence policies at other levels and thereby to improve his own objective function.

A special case of the multilevel programming problem, the linear bilevel programming problem, has been investigated extensively [2, 3, 6, 7, 15]. In such a problem, the decision variables are partitioned between high-level and low-level decision makers, each of whom has optimized his objective function. Furthermore, the decision variables in previous linear bilevel models are generally restricted to continuous variables. However, in many decentralized decision making systems, the high-level decision makers face a 'yes–no' decision problem. Therefore, we should combine the integer linear version of the program with binary restrictions on the high-level decision

* Corresponding author.

SSDI 0377-2217(94)00215-0

variables into the continuous decision variables controlled by the low-level as a mixed integer linear bilevel programming (MILBP) problem. Little research was devoted to this problem. Moore and Bard [14] gave two reasons why the MILBP problem is extremely difficult to solve, even by complete enumeration. First, it is not invariably possible to obtain a tight upper bound by using common relaxation techniques; secondly, two of three standard fathoming rules common to branch and bound can not be fully applied. Wen and Yang [16] proposed an exact algorithm to solve the MILBP; their empirical results showed that it could provide a satisfactory solution for only a relatively small problem.

In this paper, we develop a heuristic algorithm based on Tabu Search to solve the MILBP problem. The MILBP is formulated in the next section. Properties and techniques of Tabu Search are briefly described in Section 3. The proposed algorithm, and two supplementary procedures, are presented in Section 4. Computational results are provided in Section 5.

2. Problem formulation

Let $x = (y, z)$ be the vector of variables, of which y is the subvector of integer variables for the high-level problem and z is the subvector of continuous variables for the low-level problem. Then the mixed-integer linear bilevel programming problem can be expressed as that of determining $x = (y, z)$:

(MILBP)

$$\max f(x) = cy + dz \tag{1.0}$$

$$\text{s.t.} \quad y \text{ is a zero–one vector} \tag{1.1}$$

Given y satisfying (1.1), z is determined to:

$$\max g(z) = hz \tag{2.0}$$

$$s.t. \quad Ay + Bz \leqslant b, \tag{2.1}$$

$$z \geqslant 0, \tag{2.2}$$

in which $c \in \mathbb{R}^{n_1}$, d, $h \in \mathbb{R}^{n_2}$, $A \in \mathbb{R}^{m \times n_1}$, $B \in \mathbb{R}^{m \times n_2}$ and $b \in \mathbb{R}^m$.

According to this formulation, the high-level objective (1.0) is achieved by control of y, whereas the low-level objective (2.0) is achieved by control of z. Let

$$X = \{(y, z) : Ay + Bz \leqslant b,\ y \in \{0, 1\},\ z \geqslant 0\}$$

denote the problem constraint region, and

$$W_g(X) = \{(\bar{y}, \bar{z}) \in X : (\bar{y}, \bar{z}) \text{ is the optimal solution to (2) when } y \text{ is fixed at } \bar{y}\}$$

denote the feasible region of the high-level deicision maker. Without loss of generality, assume that X is nonempty and that the optimal solution of MILBP is nondegenerate. We use the following definitions to describe the feasibility and optimality.

Definition 1. A point $(\bar{y}, \bar{z})$ is said to be feasible to the MILBP if $(\bar{y}, \bar{z}) \in W_g(X)$.

Definition 2. A point (y^*, z^*) is is said to be an optimal solution to the MILBP if

(a) (y^*, z^*) is feasible; and

(b) for all feasible points $(\bar{y}, \bar{z}) \in X$,

$$cy^* + dz^* \geqslant c\bar{y} + d\bar{z}.$$

3. Tabu Search techniques

Tabu Search techniques, developed by Glover [9, 10], are primarily used to solve combinatorial optimization problems. Examples of successful applications of these techniques include graph coloring [12], traveling salesman problem [13], path assignment [1], flow shop sequencing [4, 19], job shop scheduling [8, 18], or treating learning in neural networks [5, 17].

Tabu Search is based on embedding flexible memory structures (without the type of rigidity found in branch and bound, for example) into special strategies that operate over varying time horizons, from short term to long term. A fundamental component of Tabu Search methods is a short term strategy, called 'Simple Tabu Search' in [10], which incorporates a recency-based memory. We begin by describing this component, which provides the core of our procedure.

Simple Tabu Search proceeds exactly as a local search procedure, which can be sketched as initiating from any feasible solution x' in X, a neighborhood $N(x')$ is defined for each x'. A move from x' to a neighbor x'' in $N(x')$ is then generally constructed, by optimizing an objective value of $f(x)$ over $N(x')$. This procedure is the same as a local improvement technique except for the fact that a move to a solution x'' worse than the current solution x' may be accepted. The effect of this rule is to escape from local optima; such a procedure, however, might of course cycle. A short term memory is introduced, as embedded in a list T of tabu conditions, for example; T may consist of the past k moves visited for a given k. A move remains tabu only during k iterations, so that a cyclical list T exists; whenever a move $x' \to x''$ is made, the opposite move $x'' \to x'$ is added to the end of T, with the oldest move in T being removed.

The information recorded to control the search consists of solution attributes (such as values of variables) that vary as a result of moving from one solution to another. It is generally strategically useful to maintain memory for various classes of attributes, and even for various individual attributes (as by recording for each variable the most recent iteration in which its value changed). We are not concerned with such refinements here, but observe that they have proved advantageous various settings (see, for example, [11]). An important element of Simple Tabu Search lies in the incorporation of an aspiration level $A(f(x))$, of which the value depends on a specified move and/or value of objective function f. As the searching path may be altered by the effect of T, some undesirable consequences may exist; i.e. some solutions may be forbidden that have not been visited before but would be generated from the present solution by a tabu move. In not significantly limiting the degree of freedom in the choice of a solution in $N(x')$, canceling the tabu status of a move may be acceptable; this condition would be allowed if solution x' obtained from x'' were sufficiently good. This effect can be accomplished by a situation when the objective value of x'' satisfies its aspiration condition, i.e., $f(x'') > A(f(x'))$.

To terminate the procedure, the basic step may generally be repeated until either a fixed maximum number of consecutive iterations is reached that can be performed without providing any improvement of the best value of the objective function – or until a closer estimation of the maximum value in the objective function f is achieved.

More advanced types of Tabu Search include additional types of memory, such as frequency-based memory, as a basis of strategies that extend beyond the short term. Such memory is designed particularly for intensification strategies, which seek to reinforce moves that incorporate attributes of good solutions found in the past, and diversification strategies, which seek to drive the search into unexplored regions. Records of good prior solutions are typically maintained for intensification strategies, to allow analysis of attributes common to selected groups (e.g., clusters) of such solutions, and to allow the search to revisit directly these solutions and to explore their regions more thoroughly. In our following development, we focus primarily on the short term aspects of Simple Tabu Search. These aspects provide good results for the problem setting that we examine.

4. The proposed algorithm

The process of the proposed algorithm is a succession of moves that transform a given feasible solution into an optimal or a near optimal one. The form of a move s is first specified as

$$s(y', z') = (y'', z''),$$

in which y' is a vector of zero–one high-level decision variables that can be randomly generated and z' is the optimal solution of the resultant low-level problem when the high-level variables are fixed at y', i.e. the optimal solution to the following linear program:

(PX)

$$\max\ hz$$
$$\text{s.t.}\quad Bz \leqslant b - Ay'$$
$$z \geqslant 0.$$

y'' is a vector obtained by replacing a component of y', e.g. y'_j, by $1 - y'_j$; z'' is obtained by solving (PX) with the value of high-level decision variables fixed at y''. In this setting, x'' is an adjacent point of x', and the neighborhood of x' is therefore defined as

$$N(x') = \{x'' \in X : s(x') = x''\}.$$

To prevent a move reversal, a Tabu list is formed by two sets: T0 and T1, each with a prescribed fixed length k, and both composed of the indices of the high-level decision variables. Elements of the former are the indices of the high-level decision variables that were set from 1 to 0 in the past k iterations. Elements of the latter set, in contrast, are the indices of the high-level decision variables that were set from 0 to 1 in the past k iterations. Accordingly, a move is tabu (called a tabu move) if the index of the altering variable belongs to its corresponding T0 or T1. A variable set to 0 is not allowed to be set to 1 until k other variables have been set to 0, and similary a variable set to 1 is not allowed to be set to 0 until k other variables have been set to 1.

The proposed algorithm is implemented via sequential moves. In a neighborhood search, if s is not a tabu move, then for any feasible point x' in X, x'' is selected as an adjacent point of x' such that

$$cy'' + dz'' = \max\{f(x'') : s(x') = x'',\ x'' \in N(x')\}.$$

The aspiration level $A(f(x))$, to solve the MILBP, has been set to the value of the high-level objective function of the current solution x'. When a solution x' is found, it is allowed to go to a solution x'' among $N(x')$ by a tabu move – only when the high-level objective value of x'' is larger than that of x'. Hence, canceling the Tabu status of a move may become acceptable only when it satisfies the aspiration condition. Furthermore, Tabu Search may allow for a move to a worse point (which we call a downhill move) in order to overcome local optima.

Some additional notation is introduced to provide a description of the proposed algorithm. Let $NS = n_1$ denote the total number of generated starting points, and K represent their counter. Let $ND = n_1/2$ denote the allowable number of downhill moves during a search path. The total number of neighborhood points of $x'_{[K]}$ is obviously equal to n_1; let I denote the counter to search among these neighborhood points. Therefore, we represent $y'_{[K]}$ as the vector of the high-level decision variables of the K-th generated starting point and $y''_{[I]}$ as the vector of high-level decision variables in the I-th point among neighborhood points of $x'_{[K]}$.

It is generally pointed out in Tabu Search that diversification is distinct from randomization. Diversification may proceed by periodically restarting or by periodically altering the choice rules, but in either case it seeks to assure that new solutions differ significantly from previous solutions, in ways that randomization is unlikely to achieve. (For example, diversification may specifically assign values to key variables that contrast with values that these variables have frequently received in the past, or it may systematically drive the solution a maximum distance from some "center" of previous solutions.) Some Tabu Search implementations have used randomization as a simple approximation to the goal of diversification. In our first version of Tabu Search, in which we use the Simple Tabu Search format previously described, we follow such an approach.

Simple Tabu Search algorithm

The proposed algorithm uses random restarting as an approximating basis of diversification, as follows:

Step 1. (*Initialization*)

Step 1(*A*). Set $NS = n_1$, $ND = n_1/2$ and $K = 1$.

Step 1(*B*). Set $I = 1$ and $J = 0$, if $K > NS$, go to Step 4; otherwise, go to Step 2(A).

Step 2. (*Choice*)

Step 2(*A*). Randomly generating $y'_{[K]}$, solve (PX) to obtain $(y'_{[K]}, z'_{[K]})$. If $K = 1$, compute

$$f^* = cy'_{[K]} + dz'_{[K]}.$$

Go to Step 2(B).

Step 2(*B*). If $I \leqslant n_1$, find $y''_{[I]}$ and solve (PX) to obtain $(y''_{[I]}, z''_{[I]})$ and go to step 2(C); otherwise, go to Step 2(D).

Step 2(*C*). If $x''_{[I]}$ is tabu and fails to satisfy the

aspiration condition, set $I = I + 1$, go to Step 2(B). Otherwise, record

$$\bar{f}_{[I]} = cy''_{[I]} + dz''_{[I]},$$

set $I = I + 1$, go to Step 2(B).

Step 2(D). Select the point as $(y'_{[K]}, z'_{[K]})$ with its high-level objective value being

$$\bar{\bar{f}} = \max\left\{\bar{f}_{[I]} : I = 1, 2, \ldots, n_1\right\}.$$

Step 3. (Update)

Step 3(A). Update the Tabu list T0 of T1, and the aspiration level $A(f(x))$. If $\bar{\bar{f}} \leqslant f^*$, set $J = J + 1$, go to Step 3(B). Otherwise, set $f^* = \bar{\bar{f}}$, $I = 1$ and $J = 0$, go to Step 2 (B).

Step 3(B). If $J >$ ND, set $K = K + 1$, go to Step 1(B). Otherwise, set $I = 1$, go to Step 2(B).

Step 4. (Termination)

Stop, the near-optimal solution is obtained with the high-level objective value f^*.

Two supplementary procedures

Two supplementary procedures are considered here to improve the performance of the proposed Tabu Search algorithm. The first procedure is an upperbound screening approach to decrease the computational burden of evaluation and generating neighborhood points, and the second procedure is and advanced start method to select a point to initiate the search.

At any iteration, to seek a point having the greatest high-level objective value, postoptimality pivots would be performed for all neighborhood points of the current solution. This procedure may cause excessive execution time. An upper bound for each neighborhood point is therefore developed and a potential candidate is selected according to its upper bound instead of testing all neighborhood points.

Each neighborhood point has the characteristic that all high-level decision variables are the same the current point except for one component complementary to it. We denote

$$J^+ = \left\{j \mid y_j \text{ is fixed at } 1,\ j = 1, 2., \ldots, n_1\right\}$$

and consider the following problem:

(PY)

$$\max f = \sum_{j \in J^+} c_j + \sum_{j=1}^{n_2} d_j z_j$$

$$\text{s.t.} \quad \max g = \sum_{j=1}^{n_2} h_j z_j$$

$$\text{s.t.} \quad \sum_{j=1}^{n_2} B_j z_j \leqslant b - \sum_{j \in J^+} A_j,$$

$$z_j \geqslant 0, \quad j = 1, 2, \ldots, n_2,$$

in which A_j is the j-th column vector of A, and B_j the j-th colmn vector of B.

If the low-level objective function g of (PY) is neglected, the resulting problem becomes the following linear programming problem:

(PY1)

$$\max f_1 = \sum_{j \in J^+} c_j + \sum_{j=1}^{n_2} d_j z_j$$

$$\text{s.t.} \quad \sum_{j=1}^{n_2} B_j z_j \leqslant b - \sum_{j \in J^+} A_j,$$

$$z_j \geqslant 0, \quad j = 1, 2, \ldots, n_2.$$

Let f^* be the high-level optimal objective value of (PY), f_1^* and Y^* be the optimal objective value and the dual optimal solution of (PY1) respectively. As the solution space of (PY) is contained in that of (PY1) and as the high-level objective function of (PY) is the same as the objective function of (PY1), we have $f^* \leqslant f_1^*$. Furthermore, the following problem is considered here as

(P)

$$\max Z_P = \sum_{j=1}^{n_2} d_j z_j$$

$$\text{s.t.} \quad \sum_{j=1}^{n_2} B_j z_j \leqslant b,$$

$$z_j \geqslant 0, \quad j = 1, 2, \ldots, n_2.$$

Let Z_P^* be the optimal objective value of (P) and Y_P^* be the dual optimal solution of (P). Then Y_P^* is also a dual feasible solution of (PY1); we must have

$$f_1^* - \sum_{j \in J^+} c_j = Y^*\left(b - \sum_{j \in J^+} A_j\right)$$

$$\leqslant Y_P^*\left(b - \sum_{j \in J^+} A_j\right)$$

$$= Y_P^* b - \sum_{j \in J^+} Y_P^* A_j$$

$$= Z_P^* - \sum_{j \in J^+} Y_P^* A_j,$$

or

$$f_1^* \leqslant Z_P^* + \sum_{j \in J^+} (c_j - Y_P^* A_j).$$

Hence, we found that

$$Z^U = Z_P^* + \sum_{j \in J^+} (c_j - Y_P^* Aj)$$

is an upper bound of (PY).

The postoptimality analysis is performed sequentially, after we sort the neighborhood points of the current solution according to their upper bound in descending order. Once a new point, with its high-level objective value greater than that of the current point, is obtained, we perform a move from the current point to that new point immediately. With these settings, the computational time can be moderately reduced, and the proposed algorithm is modified by replacing Step 2(B) and Step 2(C) with the following steps.

Step 2(B.1). Calculate $Z_{[I]}^{U}$, $I = 1, 2, \ldots, n_1$.

Step 2(B.2). Sequence $Z_{[I]}^{U}$, $I = 1, 2, \ldots, n_1$, in descending order. Set $I + 1$.

Step 2(B.3). If $I > n_1$, go to Step 2(D). Otherwise, find $y''_{[I]}$ and solve (PX) to obtain $(y''_{[I]}, z''_{[I]})$ and go to Step 2(C.1).

Step 2(C.1). If $x''_{[I]}$ satisfies aspiration condition, compute

$$\bar{\bar{f}} + cy''_{[I]} + dz''_{[I]},$$

go to step 3(A). Otherwise, go to Step 2(C.2).

Step (2(C.2). If $x''_{[I]}$ is tabu, set $I = I + 1$, go to Step 2(B.3). Otherwise, record

$$\bar{f} = cy''_{[I]} + dz''_{[I]},$$

set I = I + 1, go to step 2(B.3).

In the worst case of this supplementary procedure, if no neighbor solution satisfies the aspiration criterion, the algorithm behaves as before, i.e. all neighbors need to be examined for both (PX) and Z^U.

As we already observed, diversification strategies other than random restarting generally prove more effective in Tabu Search. As an alternative, we consider another option to random restarting, which executes more efficiently, based on the notion of creating an advanced start, in order to generate an initial point by more selective criteria. As a basis of this approach, we employ a heuristic algorithm for the MILBP problem proposed by Wen and Yang [16], which applied a judgment index to provide a constructed value of the high-level decision variables in its solution procedure. Empirical results showed that the heuristic algorithm performed well. We adopt a similar procedure to determine selectively a starting point.

We first define the estimated 'profit' for each high-level zero–one decision variable as

$$H(j) = c_j - Y_P^* A_j, \; j = 1, 2, \ldots, n_1,$$

in which Y_P^* is the dual optimal solution of (P). $H(j)$, in this setting, resembles reduced cost in the simplex method; hence, the greater the value of $H(j)$ is, the greater priority we assign to setting the j-th variable to one. Because the total number of generated starting points (NS) is n_1, the K-th, $K = 1, 2, \ldots, n_1$, generated starting point is then obtained by setting high-level decision variables, corresponding to $\text{IND}_{[K]}$, to one, with $\text{IND}_{[K]} = \{j |$ the indices correspond to the first K greatest values of $H(j)$, $j = 1, 2, \ldots, n_1\}$, and set the value of remaining high-level decision variable to zero. This effect can be incorporated in the proposed algorithm by replacing Step 2(A) as follows:

Step 2(A). Set the value of variables of $y'_{[K]}$, corresponding to $\text{IND}_{[K]}$, to one, and set the value of remaining variables of $y'_{[K]}$ to zero. Solve (PX) to obtain $(y'_{[K]}, z'_{[K]})$.

Set $I = 1$, $K = 1$, computing

$$f^* = cy'_{[K]} + dz'_{[K]}.$$

Go to step 2(B).

5. Computational experience

A Simple Tabu Search algorithm is proposed with two supplementary procedures, a screening

Table 1
Five different algorithms

Code	Description
1	Algorithm with full neighborhood scan and random restarting
2	Algorithm with full neighborhood scan and advanced start
3	Algorithm with screening (partial scan) and random restarting
4	Algorithm with screening (partial scan) and advanced start
5	Exact algorithm due to Wen and Yang [16]

Table 2
Problems with different problem size

Group No.	No. of constraints	No. of the high-level (0–1) variables	No. of the low-level variables
1	8	10	10
2	12	15	15
3	16	20	20
4	20	25	25

method and an advanced start method, that provide optional variations. Table 1 lists the four algorithms that can be formed using combinations of these two procedures with Tabu Search. In addition, to provide a basis for comparison, we consider an exact algorithm developed by Wen and Yang [16], denoted by Algorithm 5. In order to examine the comparative performance of the five algorithms, forty test problems were randomly generated and solved. These test problems were then separated into four groups of varied problem size (see in Table 2). All problems were solved on a CDC CYBER 840 computer. The constraint matrix coefficients of tested problems were randomly generated within a range [0, 99]. The constraint matrix had a 75% density of nonzero elements and the right hand side vector was uniformly distributed between 1/4 and 3/4 row sum of the coefficients of system constraints. All five algorithms were coded in FORTRAN V and the time limit was set to 100 CPU seconds for all tested problems.

Table 3 shows both average and standard deviation of the execution time of five algorithms. The average execution time of each algorithm has increased approximately in proportion to the problem size; except that the execution time of Algorithm 5 increased approximately exponentially. All twenty test problems of groups 3 and 4, which were run with Algorithm 5, reached the time limit. The standard deviations of the execution time of the Simple Tabu Search were relatively small, compared to those of Algorithm 5. The Tabu Search method appears to be remarkably robust in regard to the computational efficiency versus the different problem size.

Table 4 shows the average quality of the Tabu Search algorithms. The quality measure is based on the objective function value obtained by our Simple Tabu Search algorithms compared with that of Algorithm 5. The Tabu Search algorithms were found, in smaller problems of groups 1 and 2, to obtain solutions of high quality, especially in the case of Algorithm 1. The optimal solution is unknown for the relatively larger problems of groups 3 and 4, as all problems ran by Algorithm 5 reached their time limit. However, in comparison to the current solution, which is a lower

Table 3
Average execution time of algorithms

Problem group	Algorithm 1		2		3		4		5	
	$\bar{X}$	σ	$\bar{X}$	σ	$\bar{X}$	σ	$\bar{X}$	σ	$\bar{X}$	σ
1	1.22	0.22	1.07	0.31	0.83	0.49	0.88	0.17	0.45	0.48
2	4.90	1.10	4.63	1.00	3.19	1.78	3.59	0.89	26.10	32.14
3	15.62	2.62	15.37	2.96	8.01	4.40	10.70	2.02	T.L.	T.L.
4	40.50	5.84	33.03	8.20	25.21	2.94	23.29	4.74	T.L.	T.L.

Table 4
Average solution quality of Tabu Search algorithms

Problem group		Algorithm			
		1	2	3	4
1	E	100	100	96.93	100
2	E:	100	99.66	94.97	98.58
3	H:	103.97	104.10	98.90	103.52
4	H	109.43	109.82	105.57	109.47

E: optimal solution can be found by Algorithm 5 before time limit.
H: optimal solution can not be found by Algorithm 5 before time limit.

bound of the optimal solution, found by Algorithm 5 at the time limit, the average quality obtained by the Simple Tabu Search algorithms was generally found superior to that of Algorithm 5. An exception occurred for the version of the method embodied in Algorithm 3, which combined the screening (partial scan) approach with random restarting, and even in this case the quality was relatively high, and superior that of Algorithm 5 on the larger problems.

In illustrating the first supplementary procedure of applying the upperbound screening technique, average execution times of Algorithms 3 and 4 were about 70% of those of Algorithms 1 and 2. Wen applying the upperbound screening technique in seeking the best move, our observations show that the average number of points required to be solved by postoptimality analysis was approximately half the total neighborhood points. Table 4 shows that the solution quality obtained with Algorithms 3 and 4, which applied the upperbound screening technique, is a little worse than that of Algorithms 1 and 2 which did not apply this technique. The second supplementary procedure is concerned with the generation of a selective starting point in contrast to a randomly generated one. Algorithm 2 is observed in Table 3 to have performed slightly better than Algorithm 1 in terms of execution time; Algorithm 4 did not perform superior to Algorithm 3. When we observed the average solution quality in Table 4, Algorithm 2 seemed to have no significant differences from Algorithm 1, whereas Algorithm 4 expressed itself better than Algorithm 3.

6. Conclusion

In this paper we present a Simple Tabu Search algorithm to solve the mixed-integer linear bilevel programming problem. Two supplementary procedures are proposed. The first procedure is an upperbound-screening approach partially to generate neighborhood points, and the second procedure is an advanced start method to select a point to initiate the search. Compared to an exact algorithm proposed by Wen and Yang [16], all versions of our method (and three versions in particular) perform well both in terms of computational efficiency and solution quality. There is only a modest growth of computational effort required to find solutions as the problem size increases. Hence much larger problems may be successfully solved with a reasonable assurance of obtaining near-optimal or optimal solutions.

Directions for further research should be concerned with the development of Tabu Search algorithms to solve a more general case of the mixed-integer linear bilevel programming problem, e.g. proposed by Moore and Bard [14]. Both continuous and discrete variables in their problem formulation were controlled by either the high-level or the low-level decision maker. This mixed-integer linear bilevel programming problem is expected to be worth further investigation. The endeavor to develop of Tabu Search of more advanced types, such as long term memory strategy, to solve this problem should continue.

Acknowledgements

We thank the National Science Council of the Republic of China (grant NSC 81-0415-E007-08) for support, and the anonymous referees for their helpful comments and suggestions; in particular one referee made a careful scrutiny of the original version of the paper according to which we have improved the content and composition.

References

[1] Anderson, C.A., Jones, K.F., Parker, M., and Ryan, J., "Path assignment for call routing: An application of tabu

search", *Annals of Operations Research* 41 (1993) 301–312.

[2] Bard, J.F., "An algorithm for solving the general bilevel programming problem", *Mathematics of Operations Research* 8 (1983) 260–272.

[3] Bard, J.F., and Falk, J.E., "An explicit solution to the multi-level programming problem", *Computers & Operations Research* 9 (1982) 77–100.

[4] Barnes, J.W., and Laguna, M., "Solving the multiple-machine weighted flow time problem using tabu search", *IIE Transactions* 25 (1993) 121–128.

[5] Beyer, D., and Ogier, R., "Tabu learning: A neural network search method for solving nonconvex optimization problems", in: *Proceedings of the International Joint Conference on Neural Networks, IEEE and INNS,* Singapore, 1991.

[6] Bialas, W.F., and Karwan, M.H., "Two-level linear programming", *Management Science* 30 (1984) 1004–1020.

[7] Candler, W.V., and Townsley, R.J., "A linear two-level programming problem", *Computers & Operations Research* 9 (1982) 59–76.

[8] Dell'Amico, M., and Trubian, M., "Applying tabu search to the job-shop scheduling problem", *Annals of Operations Research* 41 (1993) 231–252.

[9] Glover, F., "Future paths for integer programming and links to artificial intelligence", *Computers & Operations Research* 13 (1986) 533–549.

[10] Glover, F., "Tabu Search – Part I", *ORSA Journal on Computing* 1 (1989) 190–206.

[11] Glover, F., and Laguna, M., "Tabu Search", in C.R. Reeves (ed.), *Modern Heuristic Techniques for Combinatorial Problems,* Blackwell, Oxford, 1993, 70–150.

[12] Hertz, A., and de Werra, D., "Using Tabu Search techniques for graph coloring", *Computing* 39 (1987) 345–351.

[13] Malek, M., Guruswamy, M., and Pandya, M., "Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem", *Annals of Operations Research* 21 (1989) 59–84.

[14] Moore, J.T., and Bard, J.F., "The mixed integer linear bilevel programming problem", *Operations Research* 38 (1990) 911–921.

[15] Wen, U.P., and Hsu, S.T., "Linear bi-level programming problem – A review", *Journal of the Operational Research Society* 42 (1991) 125–133.

[16] Wen, U.P., and Yang, Y.H., "Algorithms for solving the mixed integer two-level linear programming problem", *Computers & Operations Research* 17 (1990), 133–142.

[17] de Werra, D., and Hertz, A., "Tabu Search Techniques: A tutorial and an application to neural networks", *Operations Research Spectrum* 11 (1989) 131–141.

[18] Widmer, M., "Job shop scheduling with tooling constraints: A tabu search approach", *Journal of the Operational Research Society* 42 (1991) 75–82.

[19] Widmer, M., and Hertz, A., "A new heuristic method for the flow shop sequencing problem", *European Journal of Operational Research* 41 (1989) 186–193.