Pergamon

0305-0548(95)00005-4

Computers Ops Res. Vol. 23, No. 2, pp. 199–205, 1996

0305-0548/96 $9.50 + 0.00

A NOTE ON MODELING MULTIPLE CHOICE REQUIREMENTS FOR SIMPLE MIXED INTEGER PROGRAMMING SOLVERS†

Włodzimierz Ogryczak‡§

Université Libre de Bruxelles, Service de Mathématiques de la Gestion, B-1050 Brussels, Belgium

(Received November 1993; in revised form December 1994)

Scope and Purpose—In the great majority of real-life mixed integer programming models most of integer variables represent some multiple choice requirements. A multiple choice requirement is usually modeled with a generalized upper bound on a set of zero–one variables thus creating the so-called Special Ordered Set (SOS). During the past decade powerful microcomputers with friendly optimization software have become standard productivity tools for businessmen and other decision makers. Unfortunately, the branch and bound algorithms implemented there, usually, do not support the special treatment of SOS constraints. Therefore, one may face enormously long computation time while solving quite small problems including several multiple choice requirements. To overcome these difficulties we propose another algebraic model of multiple choice requirements. While using the proposed modeling technique, the standard branch and bound algorithm generates a balanced branching on the set of options included in the multiple choice requirement. It results usually in a significant shortening of the solution process.

Abstract—This note introduces the Special Ordered Inequalities (SOI) as a new technique for integer programming modeling of multiple choice requirements and piecewise linear functions. The standard modeling technique based on the use of Special Ordered Set (SOS) structure requires a branch and bound solver to be armed with the special tools for SOS processing. Otherwise, the standard branching rule applied on individual SOS variables may lead to enormously long computation process thus making impossible to solve quite small problems in a reasonable time. The proposed SOI modeling technique allows us to eliminate the necessity of the special branching rule for handling SOS structure. The standard branching on individual variables of SOI is equivalent to the special SOS branching. Thus the proposed modeling technique seems to be very attractive for users of simple mixed integer programming solvers as well as for those who want to build solver independent mathematical programming models.

1. INTRODUCTION

In the great majority of real-life mixed integer programming models most of integer variables represent some multiple choice requirements [1]. There is a variety of applications leading to models with this structure (cf. Martin and Sweeney [2]; Hummeltenberg [3] and references therein). In particular, one may consider multi-item production scheduling [4,5,6], sales resource allocation [7], menu planning [8], and catalog space planning [9]. A multiple choice requirement is usually modeled with a generalized upper bound on a set of zero–one variables [10], thus creating the so-called Special Ordered Set (SOS). For instance, the multiple choice requirement

$$z \in \{a_1, a_2, \ldots, a_r\} \tag{1}$$

†This work was partially supported by MDA Project of The International Institute for Applied Systems Analysis (IIASA), Laxenburg, Austria.

‡All correspondence should be addressed to: Włodzimierz Ogryczak, Faculty of Mathematics & Informatics, Warsaw University, Banacha 2, 02-097 Warsaw, Poland.

§Włodzimierz Ogryczak is a Visiting Professor at The Laboratory of Operations Research of U.L.B., Belgium; on leave from Faculty of Mathematics and Informatics, Warsaw University, Poland. He received both his M.S. and Ph.D. in Mathematics and Computer Science from Warsaw University. His interests focus on the computer solution to operations research and management problems, especially in the area of interactive decision support systems for multi-criteria optimization problems and linear programming solvers. He has published two books (in Polish) on computer solution to linear and integer programming problems. His papers have appeared in *Computers and Operations Research, European Journal of Operational Research, Journal of Information & Optimization Sciences, Journal of Multi-Criteria Decision Analysis, Linear Algebra and Its Applications, Mathematical Programming Study, Optimization*, and others.

where a_j represent several options (like facility capacities), may be modeled as follows

$$z = a_1x_1 + a_2x_2 + \cdots + a_rx_r$$
$$x_1 + x_2 + \cdots + x_r = 1 \tag{2}$$

$$x_j \geqslant 0, \quad x_j \quad \text{integer} \qquad \text{for} \quad j = 1,2,\ldots,r \tag{3}$$

where x_j $(j = 1,2,\ldots,r)$ are zero–one variables corresponding to several options a_j. The x_j variables with constraints (2)–(3) create the SOS which is an algebraic representation of the logical multiple choice requirement (1). If the multiple choice includes the null option $a_0 = 0$, then the corresponding variable x_o is usually omitted and the SOS constraint (2) takes the inequality form.

Problems with the SOS structure may, of course, be solved by using the standard branch and bound algorithm for mixed integer programming. However, the standard branching rule

$$x_k = 0 \quad \text{or} \quad x_k = 1 \tag{4}$$

applied on an SOS variable leads to the dichotomy

$$x_1 + x_2 + \cdots + x_{k-1} + x_{k+1} + \cdots + x_r = 1 \quad \text{or} \quad x_k = 1$$

thus creating an extremely unbalanced branching on the set of the original alternatives (any option different from a_k is selected or option a_k is selected). This causes a low effectiveness of the branch and bound algorithm. Therefore Beale and Tomlin [11] (see also Tomlin [12]) proposed a special version of the branch and bound algorithm to handle SOSs. An SOS was there treated as a single entity with branching based on the dichotomy

$$x_1 + x_2 + \cdots + x_k = 0 \quad \text{or} \quad x_{k+1} + x_{k+2} + \cdots + x_r = 0 \tag{5}$$

thus splitting the SOS into two smaller SOSs

$$x_{k+1} + x_{k+2} + \cdots + x_r = 1 \quad \text{or} \quad x_1 + x_2 + \cdots + x_k = 1.$$

It generated a complete analogy of SOS branching with direct branching on the set of multiple choice options (1)

$$z \in \{a_1, a_2, \ldots, a_k\} \quad \text{or} \quad z \in \{a_{k+1}, a_{k+2}, \ldots, a_r\}. \tag{6}$$

After development of additional techniques for large-scale problems, like pseudocosts [13], the branching rule (5) has become a standard technique implemented in mainframe mixed integer programming codes (compare, Beale [14]; Land and Powell [15]; Powell [16]; Tomlin and Welch [17]).

During the past decade powerful microcomputers have become standard productivity tools for businessmen and other decision makers. It has caused growing use of PC mathematical programming systems. They provide users with a possibility to analyze, in a convenient form, linear programs up to a couple thousands of decision variables [18]. Most systems provide also mixed integer programming capability. Unfortunately, the branch and bound algorithms implemented there, usually, do not support the special treatment of SOS constraints [19]. Therefore one may face difficulties while solving quite small problems including several multiple choice requirements. To overcome these difficulties we propose another way of modeling multiple choice requirements. While using the proposed modeling technique, the standard branching rule applied on integer variables representing the multiple choice is equivalent to the dichotomy (6) thus increasing efficiency of the branch and bound search.

2. SOI MODELING TECHNIQUE

Let us consider a multiple choice requirement modeled with the SOS (2)–(3). One may introduce new integer zero–one variables defined as the corresponding partial sums in (2), i.e.

$$y_1 = x_1$$
$$y_j = y_{j-1} + x_j \quad \text{for} \quad j = 2,3,\ldots,r.$$

Note that the standard branching on variable y_k

$$y_k = 0 \quad \text{or} \quad y_k = 1$$

implies the dichotomy (5) thus emulating the special SOS branching rule and generating a complete analogy with binary branching (6) on the set of original options.

Variables x_k no longer need to be specified as integer and, in fact, they should not be specified as integer to avoid inefficient branching on them. Moreover, they can be simply eliminated replacing the SOS model of the multiple choice (1) with the following

$$z = (a_1 - a_2)y_1 + (a_2 - a_3)y_2 + \cdots + (a_{r-1} - a_r)y_{r-1} + a_r$$
$$y_1 \leqslant y_2 \leqslant \ldots \leqslant y_{r-1} \leqslant 1 \tag{7}$$

$$y_j \geqslant 0, \quad y_j \quad \text{integer} \qquad \text{for} \quad j = 1, 2, \ldots, r-1 \tag{8}$$

where the original values of x_j are defined as the corresponding slacks in inequalities (7). Variable y_r does not arrive in (7)–(8) as by its definition it is constant and we directly put its value 1. The variables y_j with constraints (7)–(8) will be referred to as Special Ordered Inequalities (SOI).

Note that use of SOI instead of SOS does not increase the number of variables (neither integer nor continuous). In fact, any SOI contains one integer (zero–one) variable less than the corresponding SOS but it is not important as one variable could be eliminated in the latter by using inequality form for constraint (2). SOI modeling increases the number of constraints but additional constraints are very simple (network structure) and they do not cause a remarkable increase of data entries.

Example

In order to illustrate how branching would proceed with the SOI model vs how it would proceed with the SOS model, let us consider the following simple maximum problem with one multiple choice requirement in the SOS form

$$\max z$$

subject to

$$z \leqslant 1.6x_0 + 1x_1 + 2x_2 + 2x_3 + 2x_4$$
$$z \leqslant 1.6x_0 + 2x_1 + 1x_2 + 2x_3 + 2x_4$$
$$z \leqslant 1.6x_0 + 2x_1 + 2x_2 + 1x_3 + 2x_4$$
$$z \leqslant 1.6x_0 + 2x_1 + 2x_2 + 2x_3 + 1x_4$$
$$x_0 + x_1 + x_2 + x_3 + x_4 = 1$$
$$x_j \geqslant 0, \quad x_j \quad \text{integer} \qquad \text{for} \quad j = 0, 1, \ldots, 4.$$

One can easily notice that the problem has a unique optimal solution $z = 1.6$ based on $x_0 = 1$ and $x_1 = x_2 = x_3 = x_4 = 0$. However while solving the corresponding linear program we get $x_0 = 0$ and $x_1 = x_2 = x_3 = x_4 = 0.25$. Thus, with the standard branching, one of variables x_1, x_2, x_3 or x_4 is selected as the branching variable. Let say x_1 is chosen. Hence, we get two subproblems with multiple choice constraints $x_1 = 1$ and $x_0 + x_2 + x_3 + x_4 = 1$, respectively. The former has an integer solution (not optimal to the original problem). The latter has the continuous solution $x_0 = x_1 = 0$ and $x_2 = x_3 = x_4 = 0.33$ thus requiring further branching on one of variables x_2, x_3 or x_4. Finally, after examination of 5 nodes (subproblems), we conclude with the optimal integer solution.

With the SOI approach, constraints of our problem takes the following form

$$z \leqslant 0.6y_0 - y_1 + 2$$
$$z \leqslant -0.4y_0 + y_1 - y_2 + 2$$
$$z \leqslant -0.4y_0 + y_2 - y_3 + 2$$
$$z \leqslant -0.4y_0 + y_3 + 1$$

$$y_0 \leqslant y_1 \leqslant y_2 \leqslant y_3 \leqslant 1$$
$$y_j \geqslant 0, \quad y_j \quad \text{integer} \qquad \text{for} \quad j = 0,1,2,3$$

where the original variables x_j are defined as

$$x_0 = y_0$$
$$x_j = y_j - y_{j-1} \quad \text{for} \quad j = 1,2,3$$
$$x_4 = 1 - y_3.$$

The optimal solution $z = 1.6$ is now based on $y_0 = y_1 = y_2 = y_3 = 1$ and the continuous solution is $y_0 = 0$, $y_1 = 0.25$, $y_2 = 0.5$ and $y_3 = 0.75$. Thus with the standard branching strategy (maximum integer infeasibility) variable y_2 is selected as the branching one. Hence, we get two subproblems with SOI constraints $y_0 \leqslant y_1 \leqslant y_2 = y_3 = 1$ and $0 = y_0 = y_1 = y_2 \leqslant y_3 \leqslant 1$, respectively. Note that this branching is better balanced than that made in the SOS model, as the corresponding SOS constraints would take the form of $x_0 + x_1 + x_2 = 1$ and $x_3 + x_4 = 1$, respectively. One can easily notice that the first subproblem generates $y_0 = y_1 = y_2 = y_3 = 1$, which is the optimal integer solution for the original problem, whereas the second subproblem can be fathomed with this solution. Thus we complete the branch and bound process having examined 3 nodes. □

The SOI technique allows us to model easily various extensions and modifications of multiple choice requirements. For instance, in many real-life problems, one may face a multiple choice with the null option and a fixed charge connected with use of any other option. In the SOS methodology it would be modeled as follows [20]

$$x_1 + x_2 + \cdots + x_r \leqslant 1$$
$$x_j \geqslant 0, \quad x_j \quad \text{integer} \qquad \text{for} \quad j = 1,2,\ldots,r$$
$$x_j \leqslant w \quad \text{for} \quad j = 1,2,\ldots,r$$

where w is the zero–one fixed charge variable. In the corresponding SOI model

$$y_1 \leqslant y_2 \leqslant \cdots \leqslant y_r \leqslant 1$$
$$y_j \leqslant 0, \quad y_j \quad \text{integer} \qquad \text{for} \quad j = 1,2,\ldots,r$$

variable y_r can be directly used as the fixed charge variable.

3. COMPUTATIONAL EXPERIMENTS

In principle, the efficiency of the proposed modeling technique does not need any proof as it can be considered as an emulation of the SOS branch and bound algorithm [11] which efficiency has been proven in many commercial mixed integer programming systems. However, to demonstrate the importance of the use of the proposed remodeling technique, when the standard branch and bound algorithm is used, we present results of some computational experiments in Table 1.

All the problems in Table 1 are connected with the water quality management [21]. Problem size is

Table 1. Results of tests for SOI vs SOS model comparison

Problem				SOS model			SOI model		
name	m	n	c	nodes	pivots	CPU s	nodes	pivots	CPU s
t5p0	21	41	5	260	625	0.60	16	46	0.05
t5np0	21	41	5	42	84	0.10	10	52	0.07
t7p0	29	57	7	1383	3502	4.83	72	254	0.42
t7np0	29	57	7	203	473	0.65	36	181	0.26
t10p0	41	81	10	34255	86592	151.36	68	211	0.74
t10np0	41	81	10	3734	7808	16.05	84	374	0.98
t15p0	61	121	15	571603	1490064	4108.10	65534	294874	1241.32
t15np0	61	121	15	161710	351037	831.48	845	4213	11.54
t20p0	81	161	20	≫1000000	≫2510204	≫6955.24	10448	50511	297.21
t20np0	81	161	20	≫1000000	≫1857975	≫7429.50	17012	86660	363.18

described with three numbers m, n and c, where m denotes number of constraints, n number of variables, and c number of multiple choice requirements. Each multiple choice requirement covers six options (including the null option). Thus problem t5p0 contains 25 binary variables, problem t7p0—35 binary variables, etc. All the other variables are continuous and n represents the total of variables with the standard modeling of multiple choice requirements [i.e. SOS model (2)–(3)].

The problems were solved with the standard branch and bound algorithm (without special SOS handling) using LIFO node selection strategy, maximal integer infeasibility for branching variable selection and penalties on the branching variable. Thus the algorithm was quite typical for simple mixed integer programming solvers. The computation were made with the MOMIP code [22] on a DEC 5000/240 workstation. Table 1 provides total of nodes examined (subproblems solved) and total of simplex steps (pivots) during the course of the branch and bound algorithm for both SOS and SOI model. There are also given the corresponding CPU times for the branch and bound process (excluding initial LP solution). One can easily notice a dramatic improvement achieved by use of the SOI model. Improvement in the number of pivots is less than in the number of nodes since we rebuilt problems automatically without elimination of original x_j variables from the SOI model (thus increasing the linear problems size). Due to long computation we abandoned branch and bound search after examination of a million nodes. It caused that the two largest SOS models left unsolved. More precisely, in one of them the optimality proof was not completed and in one case (t20np0) the optimal solution was not even identified (the best integer solution found was about 5% worse than the optimal one). When modeled with SOI, both the problems were completely solved (with optimality proof) in less than 18,000 nodes.

Improvement in the effectiveness caused by the use of SOI model may be even greater for another branch and bound strategy. For instance, when we tried to solve the problem t10p0 with CPLEX [23], using its default strategy, it turned out to take 2796 nodes and 4157 pivots for the SOI model comparing to 734,491 nodes and 492,158 pivots for the SOS model. The latter took over 16 h (60858.90 s) of Sun Sparc 2 CPU time whereas the former took only about 1 min (64.60 s). The SOS model of t7p0 required 29,650 nodes, 19,920 pivots and 381.80 s, whereas the corresponding SOI model was solved in 5.24 s with 747 nodes and 482 pivots.

4. APPLICATION TO PIECEWISE LINEAR FUNCTIONS

Apart from direct use of multiple choice requirements, they arise also while modeling piecewise linear functions [24,10]. Suppose, we have a piecewise linear function $v = f(s)$ specified by the breakpoints

$$(s_j, v_j), \quad v_j = f(s_j) \quad \text{for} \quad j = 1, 2, \ldots, r. \tag{9}$$

Such a function is, usually, modeled on the interval $[s_1, s_r]$ as follows

$$v = \lambda_1 v_1 + \lambda_2 v_2 + \cdots + \lambda_r v_r$$

$$s = \lambda_1 s_1 + \lambda_2 s_2 + \cdots + \lambda_r s_r$$

$$\lambda_1 + \lambda_2 + \cdots + \lambda_r = 1$$

$$\lambda_j \geqslant 0 \quad \text{for} \quad j = 1, 2, \ldots, r$$

where at most two subsequent λ_j are allowed to be positive. The latter is transformed into algebraic conditions using the SOS technique

$$\lambda_1 \leqslant x_1$$

$$\lambda_j \leqslant x_{j-1} + x_j \quad \text{for} \quad j = 2, 3, \ldots, r-1$$

$$\lambda_r \leqslant x_{r-1}$$

$$x_1 + x_2 + \cdots + x_r = 1$$

$$x_j \geqslant 0, \quad x_j \quad \text{integer} \qquad \text{for} \quad j = 1, 2, \ldots, r.$$

In this formulation, SOS of variables x_j represents the multiple choice of one segment of the piecewise linear function. While applying SOS branching rule (5), one gets a partition of the entire interval $[s_1, s_r]$ into two smaller intervals

$$s \in [s_k, s_r] \quad \text{or} \quad s \in [s_1, s_k]. \tag{10}$$

But while using a simple solver with the standard branching (4) on individual variables x_k one gets

$$s \in [s_1, s_k] \cup [s_{k+1}, s_r] \quad \text{or} \quad s \in [s_k, s_{k+1}]$$

which is extremely unbalanced and thereby inefficient.

For efficient use of the standard branching rule one may remodel the SOS into the corresponding SOI as shown in the previous section. However, there is a way to get a much simpler algebraic formulation by modeling directly the piecewise linear function with SOI methodology. Namely, the piecewise linear function (9) can be modeled directly with SOI as follows

$$v = v_1 + (v_2 - v_1)u_1 + (v_3 - v_2)u_2 + \cdots + (v_r - v_{r-1})u_{r-1} \tag{11}$$

$$s = s_1 + (s_2 - s_1)u_1 + (s_3 - s_2)u_2 + \cdots + (s_r - s_{r-1})u_{r-1}$$

$$1 \geqslant u_1 \geqslant y_1 \geqslant u_2 \geqslant y_2 \geqslant \ldots \geqslant u_{r-1} \geqslant y_{r-1} \tag{12}$$

$$u_j \geqslant 0, \quad y_j \geqslant 0, \quad y_j \quad \text{integer} \qquad \text{for} \quad j = 1, 2, \ldots, r-1. \tag{13}$$

SOI (12)–(13) differs from that discussed in Section 2 [compare (7)–(8)] as only every second variable is required to be integer. Nevertheless, it keeps the most important properties of the SOI structure. The standard branching on variable y_k

$$y_k = 0 \quad \text{or} \quad y_k = 1$$

implies the dichotomy (10) thus emulating efficiency of the special SOS branching rule (5) in the SOS model of the piecewise linear function.

The SOI model of the piecewise linear function is simpler than that using SOS technique. Note that the problem to find a maximum (or minimum) of $v = f(s)$ over the interval $[s_1, s_r]$ is modeled with the SOI methodology as the problem with objective function (11) and constraints (12)–(13). The coefficient matrix of this problem is totally unimodular [23] which means that the integrality requirements on variables y_j can be simply dropped thus creating a simple linear programming model. This property is not valid for more complex problems containing more constraints on variable s. Nevertheless, it illustrates very well the simplicity and a potential power of the SOI modeling technique.

5. CONCLUDING REMARKS

An alternative technique of modeling multiple choice requirements in mathematical programming has been shown. The standard modeling technique based on the use of Special Ordered Sets, essentially, requires a branch and bound solver to be armed with the special tools for SOS processing. Otherwise, the standard branching rule applied on individual variables may lead to enormously long computation process thus making impossible to solve quite small problems in a reasonable time. It may cause difficulties with efficient use of many friendly PC optimization software tools. The proposed modeling technique of Special Ordered Inequalities allows us to eliminate the necessity of the special branching rule for handling the SOS structure. The standard branching on individual variables of SOI is equivalent to the special SOS branching. Thus the proposed modeling technique seems to be very attractive for users of simple mixed integer programming solvers as well as for those who want to build solver independent mathematical programming models.

Recently, the great progress has been made in development of languages and software supporting building of mathematical programming models. Some typical relations, like multiple choice, can be there specified by the user in a direct form leaving to the software the task of transformation into an algebraic form. For instance, a piecewise linear function can be specified by a list of breakpoints and corresponding slopes [25] while the software builds the necessary algebraic relations. The proposed

modeling techniques of SOI seems to be very attractive for a potential use in modeling software as it dramatically increases the solver independence of the generated model.

SOI modeling technique seems to be very interesting in connection with recently developed mixed integer programming algorithms using the so-called convexification procedures ([26] and references therein). This requires, however, further research and extensive computational tests.

REFERENCES

1. W. C. Healy, Multiple choice programming. *Ops Res.* **12,** 122–138 (1964).
2. R. K. Martin and D. J. Sweeney, An ideal column algorithm for integer programs with special ordered sets of variables. *Mathl Prog.* **26,** 48–63 (1983).
3. W. Hummeltenberg, Implementations of special ordered sets in MP software. *Eur. J. Opl Res.* **17,** 1–15 (1984).
4. L. S. Lasdon and R. C. Terjung, An efficient algorithm for multi-item scheduling, *Ops Res.* **19,** 998–1022 (1971).
5. A. A. B. Pritsker, L. J. Waters and P. M. Wolfe, Multiproject scheduling with limited resources: A zero–one programming approach. *Mgmt Sci.* **16,** 93–108 (1969).
6. D. J. Sweeney and R. A. Murphy, Branch and bound methods for multi-item scheduling. *Ops Res.* **29,** 853–864 (1981).
7. A. A. Zoltners and P. Sinha, Integer programming models for sales resource allocation. *Mgmt Sci.* **26,** 242–260 (1980).
8. J. L. Balintfy, Menu planning by computer. *The Commun. of ACM* **7,** 255–259 (1964).
9. M. Johnson, A. Zoltners and P. Sinha, An allocation model for catalog space planning. *Mgmt Sci.* **25,** 117–129 (1979).
10. H. P. Williams, *Model Building in Mathematical Programming* (Third edition). Wiley, New York (1991).
11. E. M. L. Beale and J. A. Tomlin, Special facilities in a general Mathematical programming system for non-convex problems using ordered sets of variables. In J. Lawrence (Ed) *Proc. 5th IFORS Conference* (pp. 447–454). Tavistock, London (1970).
12. J. A. Tomlin, Branch and bound methods for integer and nonconvex programming. In J. Abadie (Ed) *Integer and Nonlinear Programming* (pp. 437–450). North-Holland, Amsterdam (1970).
13. J. J. H. Forrest, J. P. H. Hirst and J. A. Tomlin, Practical solution of large mixed integer programming problems with UMPIRE. *Mgmt Sci.* **20,** 736–773 (1974).
14. E. M. L. Beale, Branch and bound methods for mathematical programming systems. In P. L. Hammer, E. L. Johnson and B. H. Korte (Eds) *Annals of Discrete Mathematics 5: Discrete Optimization* (pp. 201–219). North-Holland, Amsterdam (1979).
15. A. H. Land and S. Powell, Computer codes for problems of integer programming. In P. L. Hammer E. L. Johnson and B. H. Korte (Eds) *Annals of Discrete Mathematics 5: Discrete Optimization* (pp. 221–269). North-Holland, Amsterdam (1979).
16. S. Powell, Software. In M. O'hEigertaigh, J. K. Lenstra and A. H. G. Rinnooy Kan (Eds), *Combinatorial Optimization: Annotated Bibliographies* (pp. 190–194). Wiley, New York (1985).
17. J. A. Tomlin and J. S. Welch, Mathematical Programming Systems. In E. Coffman and J. K. Lenstra (Eds) *Handbook of Operations Research and Management Science: Computation.* North-Holland, Amsterdam (1993).
18. R. Sharda, Linear programming software for personal computers: 1992 survey. *OR/MS Today*, 44–60 (June 1992).
19. M. J. Saltzman, Survey: mixed-integer programming. *OR/MS Today* 42–51 (April 1994).
20. E. L. Johnson, M. M. Kostreva and U. H. Suhl, Solving 0–1 integer problems arising from large scale planning models. *Ops Res.* **33,** 803–819 (1985).
21. R. Berkemer, M. Makowski and D. Watkins, A prototype of a decision support system for water quality management in central and eastern Europe, WP-93-049. IIASA, Laxenburg.
22. W. Ogryczak and K. Zorychata, Modular optimizer for mixed integer programming—MOMIP version 1.1, WP-93-055. IIASA, Laxenburg (1993).
23. CPLEX Optimization, Using the CPLEX Callable Library and CPLEX Mixed Integer Library, CPLEX Optimization, Incline Village (1993).
24. G. L. Nembauser and L. A. Wolsey, *Integer and Combinatorial Optimization.* Wiley, New York (1988).
25. R. Fourer and D. M. Gay, Expressing special structures in an algebraic modeling language for mathematical programming. Technical Report 91–09, Northwestern University, Evanston, (1993).
26. E. Balas, S. Ceria and G. Cornuejols, A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathl Prog.* **58,** 295–324 (1993).