Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A Note on the Additive Algorithm of Balas

Fred Glover, Stanley Zionts,

33. F. RADÓ, "Linear Programming with Logical Conditions" (in Rumanian), *Comunicările Academiei RPR* **13**, 1039–1041 (1963).
34. J. D. C. LITTLE, K. G. MURTY, D. W. SWEENEY, AND C. KAREL, "An Algorithm for the Traveling Salesman Problem," *Opns. Res.* **11**, 972–989 (1963).
35. P. BERTHIER AND PH. T. NGHIEM, "Résolution de problèmes en variables bivalentes (Algorithme de Balas et procédure S.E.P.). Note de travail no. 33, Société d'Économie et de Mathématique Appliquées," Paris, 1965.

A NOTE ON THE ADDITIVE ALGORITHM OF BALAS†

Fred Glover

Carnegie Institute of Technology, Pittsburgh, Pa.

and

Stanley Zionts

Carnegie Institute of Technology and U. S. Steel Corp. Applied Research Laboratory, Monroeville, Pa.

(Received December 28, 1964)

IN THE preceding paper EGON BALAS presents an interesting combinatorial approach to solving linear programs with zero-one variables. The method is essentially a tree-search algorithm that uses information generated in the search to exclude portions of the tree from consideration. The purpose of this note is: (1) to propose additional tests‡ meant to increase the power of Balas' algorithm by reducing the number of possible solutions examined in the course of computation; and (2) to propose an application for which the algorithm appears particularly well-suited. An acquaintance with Balas' paper is assumed in the discussion that follows.

First consider ways of reducing the number of solutions examined by Balas' method. In one approach to this objective, Balas defines the set D_s that (using his notation and equation numbers) consists of those $j\epsilon(N-C^s)$ such that, if $\boldsymbol{a}_j$ were introduced into the basis, the value of the objective function would equal or exceed the best value ($z^{*(s)}$) already obtained. It then follows immediately that the variables associated with elements of D_s may be ignored in seeking an improving solution along the current branch of the solution tree. However, D_s can be fruitfully enlarged by using a slightly less immediate criterion for inclusion, specified as follows. Each j in N_s is first examined—in any desired sequence—to see if

† The authors originally refereed Mr. Balas' paper for *Operations Research*. With their referee's report they submitted this manuscript extending some of his results.

‡ The additional tests proposed in this note reduce the number of solutions to be examined under the additive algorithm, at the expense of an increase in the amount of computation at each iteration. While we conjecture that on the balance it is worthwhile introducing these tests, this is a matter to be decided on the basis of computational experience.

$J_s \cup \{j\}$ gives a feasible solution. (Such an examination will sometime have to be made for a number of these j in any case.) If feasibility is established for any $J_s \cup \{j\}$, a new improving solution is obtained that may be handled in the usual fashion by the algorithm. But in the more probable event that $J_s \cup \{j\}$ fails to satisfy one of the constraints—say constraint $\bar{i}(a_{\bar{i}j} > y_{\bar{i}}^s)$—consider the objective function coefficient $c_h = \min[c_p \mid p \epsilon (N_s - \{j\})$ and $a_{\bar{i}p} < 0]$. By referring to Balas' definition of N_s it can readily be seen that D_s may be enlarged to include j if the sum $c_h + c_j$, when added to the current objective function value z_s, will produce a value at least as great as $z^{*(s)}$.† Similarly, we may enlarge the set D_k^s in a corresponding manner to reduce the number of variables that need to be considered after backtracking with the algorithm.

Another means that can be employed to sharpen the efficiency of the algorithm is to include an additional relation at Steps 3 and 6 of the algorithm. To understand the function of these proposed relations, it is useful to review the relations (22) and (27) that Balas includes at these steps. For any unsatisfied constraint (y_i^s or $y_i^k < 0$), the additive algorithm instructs by means of (22) and (27) to determine whether that constraint can be satisfied by setting all nonexcluded variables with negative coefficients equal to one. If not, since the constraint represents a 'less than or equal to' relation, obviously there is no improving solution that is feasible along the present branch of the tree, and the algorithm backtracks appropriately. But we note that it is entirely possible for the tests embodied in (22) and (27) to be passed only at the expense of forcing the objective function too high. Restricting attention for the moment to Step 3, we specify a sufficient (but not necessary) condition to detect whether such a situation obtains. If there exists an i such that $y_i^s < 0$ and $y_i^s(c_j/a_{\bar{i}j}) \geqq z^{*(s)} - z_s$ for all $j \epsilon N_s(a_{\bar{i}j} < 0)$, then there are no feasible solutions left along the current branch of the tree that will improve on the best already found. To prove this we assume that an improving solution does exist. Then for any such solution we have $\sum a_{\bar{i}j} \leqq y_i^s$, where the sum is taken over those negative a_{ij} such that $j \epsilon N_s$ and $\boldsymbol{a}_j$ is in the basis for the specified solution. Then if $\sum c_j$ is restricted to these same j, the current objective function value z_s plus $\sum c_j$ yields a lower bound on the objective function value of the improving solution. But by the foregoing we have $z_s + \sum c_j = z_s + \sum a_{\bar{i}j}(c_j/a_{\bar{i}j}) \geqq z_s + \sum a_{\bar{i}j}(z^{*(s)} - z_s)/y_i^s \geqq z^{*(s)}$, and the assertion is proved by contradiction. Similar inferences may be drawn relative to the conditions encountered at Step 6. The wording of the expanded instruction at this step may then be: "For those i such that $y_i^k < 0$, check the relation (27a) $y_i^k(c_j/a_{\bar{i}j}) \geqq z^{*(s)} - z_k$ for all $j \epsilon N_k^s$, and for $k = k_1$. If (27a) holds for any i, carry the procedure indicated when (27) does not hold for some i."‡

† If c_h is undefined, it is the same as failing to satisfy relation (22) in step 3 of the additive algorithm which we discuss below. Broader criteria for inclusion in D_s may clearly be established by considering whether $J_s \cup \{j\} \cup \{h\}$ is feasible, but at a computational cost that probably outweighs the gains to be derived. In a contrasting vein, a more expedient, but less restrictive approach would be to use 1 in place of c_h, provided the c_p consist of positive integers.

‡ For a more general test which becomes quite powerful in a somewhat different combinatorial approach than employed by Balas, *see* reference 5. For applications of nonbinding constraint elimination and extraneous variable elimination to mathematical programming in general, *see* reference 6.

We turn now to the issue of generating and searching the solution tree. First we note that for many problems a feasible solution is known in advance. This solution may be used to establish a starting value for $z^{*(s)}$ other than infinity, thereby expediting convergence to the optimum. But such a solution may also be used in another way. If it is suspected that an appreciable fraction of the variables for the feasible solution coincide in value to those for some optimal solution, it may be useful to employ a two-stage algorithm that treats the ('primary') variables set at unity in the feasible solution differently from the ('secondary') variables set equal to zero. In particular, it would seem desirable for such an algorithm to be designed to dispose rapidly of various 0–1 assignments to the primary variables in the first stage and then to apply the additive algorithm to the secondary variables in the second stage. Though it is beyond the scope of this note to describe such a procedure in detail, we remark that it is possible to use a simplified bookkeeping scheme for the first stage (consisting of a single vector each of whose components assumes only three values) so that with slight modifications the tests described above may still be applied to restrict the range of 0–1 assignments necessary for consideration (*see* reference 5).

An application for the Balas' algorithm of interest lies in its potential integration with the GILMORE-GOMORY method for solving the cutting-stock problem.[3, 4] While this problem may be given an integer programming formulation, the number of variables, even for a moderate-sized problem, is so large that, in a practical sense, the usual integer programming approach is to no avail. Even the ordinary linear programming methods are not practical. Gilmore and Gomory's method overcomes this difficulty by restricting attention to a very small subset of the variables in order to obtain a starting solution via linear programming. Once a solution is available, a solution to the knapsack problem is used to generate improving variables for the problem.† The solution process then alternates between generating new variables and solving the linear programming problem to see which of the newly generated variables should be included in the solution. When no improving variables can be found the algorithm comes to a halt.

Three features of Balas' method should prove useful in this application. First, Gilmore and Gomory's method does not provide integer solutions; Balas' algorithm appears very reasonable in this context. Second, except for the first solution of the integer-program subproblem, a finite starting $z^{*(s)}$ (obtained from the previous subproblem) is available. Such solutions can be exploited as suggested above. Third, at any stage of the Gilmore-Gomory method it is necessary to consider only those solutions involving at least one of the newly generated variables, a situation with which Balas' method dovetails rather well.

A significant portion of the cutting-stock problems known to us can be represented using zero-one variables, higher-valued integers of course being represented by sums of zero-one integers. Balas' extension of the additive algorithm to the general integer linear programming problem[2] proposes an improvement in efficiency of such a representation that makes the approach even more attractive.

† In the first version of their cutting-stock method, Gilmore and Gomory solve the knapsack problem by dynamic programming; in the second, they propose a special algorithm for the knapsack problem which proves to be substantially superior to dynamic programming.

REFERENCES

1. EGON BALAS, "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," *Opns. Res.* **13**, 517–546 (1965).
2. ———, "Extension de l'algorithme additif a la programmation en nombres entiers et a la programmation nonlineaire," *Comptes Rendus de l'Academie des Sciences* (Paris) **258**, 5136–5139 (1964).
3. P. C. GILMORE AND R. E. GOMORY, "A Linear Programming Approach to the Cutting-Stock Problem," *Opns. Res.* **9**, 849–859 (1961).
4. ——— AND ———, "A Linear Programming Approach to the Cutting-Stock Problem—Part II," *Opns. Res.* **11**, 863–888 (1963).
5. FRED GLOVER, "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem," (forthcoming).
6. S. ZIONTS, G. L. THOMPSON, AND F. M. TONGE, "Techniques for Removing Nonbinding Constraints and Extraneous Variables from Linear Programming Problems," Carnegie Institute of Technology, Graduate School of Industrial Administration, Pittsburgh, Pennsylvania, November, 1964.