OMEGA, The Int. Jl of Mgmt Sci., Vol. 4, No. 1, 1976. Pergamon Press. Printed in Great Britain

A Guided Tour of Recent Practical Advances in Integer Linear Programming

AM GEOFFRION

(*Received September 1974; in revised form May 1975*)

The field of integer programming has been an extremely active one in recent years. Only a small fraction of the contributions, however, are of demonstrable *practical* value to practitioners or software designers within the near future. These are the contributions which this paper will attempt to survey. The selections are made on a subjective basis from the published and unpublished literature appearing over the last few years.[1]

1. LINEAR PROGRAMMING ADVANCES USEFUL FOR INTEGER PROGRAMMING

SINCE linear programming (LP) provides the main computational horsepower for the leading approaches to integer linear programming (IP) (especially branch-and-bound), almost any computational advance in LP is automatically beneficial to IP. The well-known technique for generalized upper bounding (GUB), for instance, has proven valuable in IP because so many IP models contain constraints of this form. The same will probably be true of the new generation of in-core LP optimizers.

This point applies particularly to the field of large-scale linear programming for the obvious reasons and also, as indicated in more detail below, because integer programs of moderate size can enlarge impressively when seemingly redundant constraints are added for the purpose of obtaining stronger bounds. Tomlin [39] has presented a brief but incisive review of significant recent developments in large-scale LP. Another excellent review, more general in nature and with more emphasis on practical software considerations, is provided by White [40]. More recently, improved representations of the inverse have been found that permit substantially reduced pivot times and storage requirements for various specially structured large-scale problems [21, 26, 43].

Computational techniques for solving transportation and network problems

[1] Earlier algorithmic progress was surveyed in Ref. [15].

have advanced markedly. The use of better information structures has led to greatly improved specialized implementations of the primal simplex method for such problems. Several new implementations have proven clearly superior to the venerable out-of-kilter method [5, 18, 20, 23, 29]. It appears likely at this time that some of the updating techniques used in these incredibly efficient implementations admit algebraic generalizations to quite general non-network LP structures. This could have a major influence on the future of large-scale linear programming technology. In the meantime, and in any event, these network implementations can be used as the heart of branch-and-bound algorithms for a wide variety of integer programming problems that can be viewed in terms of networks.

The recent development of interactive implementations of full scale mathematical programming systems is of potentially great significance [7, 19, 25]. It should be obvious that the conveniences possible with an interactive implementation can enhance a user's productivity and overall effectiveness in almost every phase of his activities—data management, modeling, problem modification, run control, solution access and analysis, and algorithmic experimentation. The obstacles to truly interactive problem solving can be substantially diminished, a particularly important advantage in view of the fact that completely automated solution procedures for most complex problems do not exist now and never will. These advantages are especially valuable in the context of integer programming, where the principles of effective modeling and algorithmic tactics are much less routine and more in need of experimentation and closely supervised control than for linear programming. A successful practical application is described by Thomas *et al.* [38].

2. MODELING PRINCIPLES

The computational tractability of any given IP application is strongly dependent on both the *content* (assumptions) of the model and the way in which the model is *represented* mathematically (the distinction here is important). It is essential to recognize that some of the guiding principles from linear programming can be downright dangerous if applied absent-mindedly to integer programs.

Modelers should be aware that integer programming is susceptible to a pathology that rarely arises in linear programming: possible discontinuity of the optimal value as a function of constraint coefficients. If a small change in some 'soft' coefficient leads to a sudden incommensurately large decrease in the optimal cost, the analyst will very likely want to revise this coefficient or perhaps even revise the model so as to treat this coefficient as an explicit decision variable. This pathology is probably much more frequent than is commonly realized. Surprisingly, very few papers have ever addressed this topic seriously. Williams' [41] paper was among the first. Some theoretical results regarding continuity

in the IP context are established and penalty functions are recommended as a remedy for constraints whose data are likely to allow troublesome discontinuities. More comprehensive results are given by Radke [33]. Armstrong and Sinha [1] have presented an impressive modeling lesson very closely related to the discontinuity pathology. Also, Geoffrion and Graves [14] have found that minor adjustment to the problem statement can greatly mitigate the bogey of discontinuity.

Let us turn now to the topic of model representation for computational effectiveness. Discussion will largely be limited to the context of LP-based branch-and-bound, as virtually all commercially available IP software is of this type. The main lesson of Williams [42] is an important one: one should examine the various possible mathematical representations of a model which are equivalent in a logical sense and select the one which seems likely to give the tightest bound when relaxed in the usual way to an ordinary LP. The reason is that better bounds imply less need for branching, thereby shifting the balance of work to the relatively more efficient machinery of linear programming (as opposed to enumeration). Williams [42] gives five specific examples to illustrate various ways of achieving 'equivalent' formulations yielding better LP bounds. His first example shows that constraints can sometimes be made tighter without encroaching on the region of integer feasibility by adjusting coefficients systematically to exploit the integral nature of some of the variables. This point is also made forcefully by Sommer [37]. I have observed the value of this simple idea on several occasions; e.g. in the context of avoiding unrealistically large capacity limits in facility location problems.

Williams' other examples illustrate instances in which new constraints can be added that are redundant in an IP sense but are not so for the associated LP relaxation. In the second example these constraints can be discovered by graphically examining two- or three-dimensional components of the problem. For the remaining examples they can be discovered by 'disaggregating' existing constraints, as by writing $x_1 \leqslant x_4$, $x_2 \leqslant x_4$, and $x_3 \leqslant x_4$ instead of $x_1 + x_2 + x_3 \leqslant 3x_4$ when the variables are 0–1. A technique very close in spirit to constraint disaggregation, especially for problems with a preponderance of 0–1 variables, is the determination of simple logical relationships between 0–1 variables which are not already expressed by correspondingly simple constraints; e.g. if $x_1 = 1$ implies $x_2 = 1$, then one may introduce $x_2 \geq x_1$. Hammer and Nguyen [22] have described a simple procedure for systematically ferreting out such information.

Another technique for generating useful 'redundant' constraints is to explicitly derive the convex hull of a select (and relatively simple) subset of the set of all constraints. An illustration of this technique has been presented by Geoffrion and McBride [16].

Additional constraints discovered by one or another of the means described above can be used in a number of other ways beyond simply appending them

to the LP relaxation for bounding; for instance, they can be used to quickly eliminate variables at various points in the search tree, and to strengthen so-called penalties. In addition to [42], confirmation of their value for large practical applications can be found in Refs. [16] and [13, see especially Section 5], and in Ref. [4].

Thus, the integer programming modeler must learn that economizing on the number of constraints in the representation of a model can be a sin rather than a virtue. Economizing on the number of integer variables, however, is usually very desirable. New tricks are found from time to time which permit more compact transformations of combinatorial problems or nonlinear integer programs into linear integer programs, usually with the help of binary variables. For instance, an improved technique for linearizing quadratic forms in n binary variables by introducing only n continuous variables and $4n$ constraints is given in [17]. Having said this, a word of caution must be added that it is often very inefficient to transform an inherently combinatorial or nonlinear problem into a linear integer program. One should also keep in mind that there are conspicuous exceptions to the rule that the number of integer variables should be minimized. Sommer [37] argues convincingly (see Cases 1 and 4) that important decision alternatives should always be represented explicitly by individual integer variables rather than implicitly in terms of several variables, even if this means introducing some additional binary variables. The reason has to do with the fact that available IP codes can branch only on explicit integer variables (although many other kinds of branching are possible); and it is usually preferable to branch on the 'most important' decision alternatives early in the branch-and-bound search.

3. ALGORITHMS

Many intriguing algorithmic ideas continue to be proposed or developed in the literature, but relatively few are put to serious computational test. It is therefore difficult to know which developments are of truly practical value. In the near term, at least, it is probably safe to limit attention to contributions that are capable of enhancing the acknowledged general supremacy of the LP-based branch-and-bound approach.

A great deal remains to be learned about utilizing the broad tactical flexibility inherent in LP-based branch-and-bound. An important paper containing a wealth of ideas and computational evidence on penalties, priorities, branching criteria, 'pseudocosts', and related topics is by Forrest *et al.* [11]. Other substantial computational studies in a similar vein are by Mitra [28], Breu and Burdet [6] and Piper [32].

It would appear that there are two distinct kinds of evolution going on as regards the current generation of commercial software. One is a pragmatic

gravitation toward the incorporation and regular use of improved tactical options (e.g. better 'pseudocosts') for general mixed integer linear programs. This kind of gradual evolution, while an essential part of the process of assimilating a new commercial technology on a widespread basis, has probably reached a stage of diminishing returns. Further substantial increases in computational effectiveness will probably depend primarily on a different kind of evolution: innovations for the strategic exploitation of special problem structure. The outstanding example of this is the treatment of 'special ordered sets' by UMPIRE (see [11] and the references therein). The first type of special ordered set deals with multiple choice constraints over 0–1 variables, while the second deals with the structure that arises when economies-of-scale and other non-linearities are treated to piecewise-linear approximation. Both of these structures occur in a wide variety of practical problems, and have been effectively exploited so as to yield substantially improved computational efficiency. Beale and Forrest [3] have made a noteworthy contribution in this vein. It is shown how automatic interpolation can be carried out so as to avoid having to preselect the piecewise-linear approximation with the second type of special ordered set.

Another strategic development of the second evolutionary kind is Lagrangean Relaxation, which can improve the standard bounds available from linear programming (see Geoffrion [12]). It is based on a formal application of Lagrangean duality ideas and the empirical observation that the 'duality gap' tends to be very small in many applications. This technique can exploit a variety of special but common types of constraints so as to produce substantially improved bounds, penalties and cutting-planes at modest computational cost. For instance, it was found in [16] for four real facility location problems that Lagrangean Relaxation bridged an average of 60 per cent of the gap between the LP optimal value and the integer optimal value for a nominal computational cost, with a maximum average potential of 97 per cent if greater effort is expended. Moreover, the penalties associated with fractional variables were one or two orders of magnitude better than those available from standard formulae yet were less expensive to compute. Of course such favorable results may not be obtained for other applications, but independent results by other investigators for other problems are encouraging. Lagrangean Relaxation has been used successfully by M Held and RM Karp in their well-known and successful attack on the traveling salesman problem. Another nice illustration is provided by Ross and Soland [36]. They have succeeded in solving highly structured problems with several thousand binary variables in a few seconds on a large machine. Still another successful application is reported by Fisher [9]. See also the ambitious paper by Fisher *et al.*[10]. It is interesting to note that these other investigators have all used Lagrangean Relaxation *alone*, rather than as an adjunct to linear programming. It can be used either way.

A point made earlier in these notes was that improved bounds are frequently possible through improved model representation practices. My impression is

that there is a lot of room for improvement here. Some of the tricks mentioned in that context could even be automated as part of the preprocessing of a problem prior to its solution. To the extent that this leads to the introduction of additional constraints, it reminds one of the not infrequently mentioned suggestion to incorporate cutting-planes into branch-and-bound as a means of improving the tightness of the usual LP relaxation, and to facilitate effective branching (cf. Section IV of [15]). Many varieties of cutting-planes have been proposed over the years, the vast majority of them untested, but excellent results have been obtained using just the original Gomory mixed integer cuts of 1960 in a hybrid branch-and-bound/cutting-plane implementation for solving the master problems in [13]. Very few studies of this type of hybrid algorithm have been reported. One modest study by Knuts [24] has appeared on a hybrid version of Gomory cutting-planes with Balas' algorithm. The hybrid algorithm proved superior to the pure algorithms on which it was based.

More than a decade ago, JF Benders proposed an approach for iteratively decomposing mixed integer problems into their continuous and integer parts, each of which can then be solved by appropriate specialized algorithms. Progress has been made recently in finding variants of this approach which work well for specific classes of problems. A large scale implementation and its successful application to a practical problem is described by Geoffrion and Graves [13]. An interesting point discussed in [13] is that the choice of the mathematical representation of a model can have a decisive influence on computational efficiency; moreover, the most powerful representation for this class of problem is too unwieldy to be attacked directly via branch-and-bound. Other promising variants of Benders' approach are reported in [8] and [27]. In addition, a successful practical largescale application using Benders' decomposition for the LP relaxations within branch-and-bound is described by Austin and Hogan [2].

IV. *Post-optimal and reoptimization capability*

It is surprising that so little attention has been devoted to post-optimal analysis and reoptimization capabilities for integer programming. These topics, so well developed and widely used in the domain of linear programming, are needed even more acutely for practical applications of IP models. This is because the intrinsic nature of integer programming generally calls for greater caution in interpreting the results and generally requires more runs in order to do justice to a real application, especially strategic applications. For instance, [13] identifies and discusses eight types of runs necessary to properly carry out a facility location study. Most of these actually call for an entire sequence of related runs, each of which must be solved to a very high degree of optimality in order to be meaningful (since it is the *differences* between the solutions of the runs in a series that matters). Thus there is a clear need for true reoptimization and post-optimal capability.

The first attempt to build such capability into a branch-and-bound algorithm

was carried out by Roodman [34] in the context of Balas' additive algorithm. The basic concept in [34] is that for the purposes of postoptimization it is sufficient to limit consideration to the partial solutions that were fathomed in the course of the initial solution of the problem. Similar developments in the more pertinent context of LP-based branch-and-bound are given in [35]. A relatively comprehensive study of this budding field has just been completed by Nauss [30]. It includes extensive computational experience with several classes of problems, some of it of independent interest as further experimental confirmation of the value of the Lagrangean relaxation ideas mentioned earlier.

Benders' decomposition method presents a different kind of opportunity for reoptimization: the so-called cuts generated to solve one problem can be saved and modified at little cost so as to be useable in many modified versions of the same problem. This point is discussed in detail in Section 2.4 of [13] and has also been recognized and employed successfully (60–80 per cent reduction in computing time) by Noonan [31]. With a little care, a number of cutting-plane algorithms also lend themselves to reoptimization, although no computational experience has yet been reported.

ACKNOWLEDGEMENTS

Partially supported by the National Science Foundation under Grant GP-36090X and by the Office of Naval Research under Contract N00014-69-A-0200-4042. The author would like to express his appreciation to David C Sommer and William W White for their valuable comments on an earlier version of this note. Reproduction in whole or in part is permitted for any purpose of the United States Government.

REFERENCES

1. Armstrong RD and Sinha P (1974) Application of quasi-integer programming to the solution of menu planning problems with variable portion size. *Mgmt Sci.* **21**(4).
2. Austin LM and Hogan WW Optimizing procurement of aviation fuels. *Mgmt Sci.* forthcoming.
3. Beale EML and Forrest JJ (1973) Global optimization using special ordered sets. Scientific Control Systems Ltd.
4. Beale EML and Tomlin JA (1972) An integer programming approach to a class of combinatorial problems. *Math. Progm.* 3(3), 339–344.
5. Bradley GH, Brown GG and Graves GW (1975) A comparison of storage structures for primal network codes. Paper presented at the ORSA/TIMS National Meeting, Chicago, May.
6. Breu R and Burdet C (1974) Branch and bound experiments in zero-one programming. *Math. Progm. Study* 2.
7. Computer Research Center for Economics and Management Science (1974) SESAME: design and capabilities overview. Document D0075, National Bureau of Economic Research, Inc., Cambridge, Mass. March. (Public domain)

8. ETSCHMAIER MM and RICHARDSON RJ (1973) Improving the efficiency of Benders' decomposition algorithm for a special problem in transportation. Tech. Rep. No. 14, Dep. of Industrial Engineering, University of Pittsburgh, May.
9. FISHER ML (1974) A dual algorithm for the one-machine scheduling problem. Tech. Rep. No. 243, Dep. of Operations Research, Cornell University, December.
0. FISHER ML, NORTHUP WD and SHAPIRO JF (1974) Using duality to solve discrete optimization problems: theory and computational experience. Rep. 7409, Graduate School of Business, University of Chicago, February. *Math. Progm.* forthcoming.
11. FORREST JJH, HIRST JPH and TOMLIN JA (1974) Practical solution of large mixed integer programming problems with UMPIRE. *Mgmt Sci.* **20**(5), 736–773.
12. GEOFFRION, AM (1974) Lagrangean relaxation for integer programming. *Math. Progm. Study* 2, 82–114.
13. GEOFFRION AM and GRAVES GW (1974) Multicommodity distribution system design by Benders decomposition. *Mgmt Sci.* **20** (5), 822–844.
14. GEOFFRION AM and GRAVES GW (1975) Scheduling parallel production lines with changeover costs: practical application of a quadratic assignment/LP approach. Working Paper No. 231, Western Management Science Institute, UCLA, April.
15. GEOFFRION AM and MARSTEN RE (1972) Integer programming algorithms: a framework and state-of-the-art survey. *Mgmt Sci.* **18**(7), 465–491.
16. GEOFFRION AM and MCBRIDE RD (1972) The capacitated facility location problem with additional constraints. Paper presented at the Joint National Meeting of AIIE, ORSA and TIMS, Atlantic City, N.J. November.
17. GLOVER F (1974) Improved linear integer programming formulations of nonlinear integer programs. *Mgmt Sci. Rep.* No. 72–8, School of Business, University of Colorado, Rev. April. *Mgmt Sci.* forthcoming.
18. GLOVER F, KARNEY D, KLINGMAN D and NAPIER A (1974) A computation study on start procedures, basis change criteria, and solution algorithms for transportation problems. *Mgmt Sci.* **20**(5), 793–813.
19. GONCHARSKY RS, RAUCH A and WHITE WW (1973) Large scale mathematical programming in an APL environment. Tech. Rep. No. 320–3027, IBM Philadelphia Scientific Center, October. (Program is not available outside of IBM.)
20. GRAVES GW and MCBRIDE RD (1973) The use of inherent triangularity in network problems. Paper presented at the 44th National Meeting of ORSA, San Diego, Calif. November.
21. GRAVES GW and MCBRIDE RD (1975) The factorization approach to large-scale linear programming. Working Paper No. 208, Western Management Science Institute, UCLA, Rev. May. *Math. Progm.* forthcoming.
22. HAMMER PL and NGUYEN S (1972) APOSS: a partial order in the solution space of bivalent programs. C.R.M.-163, Centre de Recherches Mathematiques, Universite de Montreal, April.
23. KARNEY D and KLINGMAN D (1973) Implementation and computational study of an incore out-of-core primal network code. Rep. C.S. 158, Center for Cybernetic Studies, University of Texas, Austin, October. *Ops Res.* forthcoming.
24. KNUTS L (1973) Combining two integer programming algorithms. *Acta Acad. abo.* Ser. B, **33**(12), 1–8.
25. MANAGEMENT SCIENCE SYSTEMS (1974) MPS III/OL user manual. Rockville, Maryland. (Proprietary system)
26. MCBRIDE RD (1973) Factorization in large-scale linear programming. PhD Dissertation University of California, Los Angeles, June.
27. MCDANIEL D and DEVINE M (1974) Alternative relaxation schemes for Benders' partitioning approach to mixed integer programming. School of Industrial Engineering, University of Oklahoma, April.
28. MITRA G (1973) Investigation of some branch-and-bound strategies for the solution of mixed integer linear programs. *Math. Progm.* **4**(2), 155–170.
29. MULVEY JM (1975) Developing and testing a truly large-scale network optimization code. Paper presented at the ORSA/TIMS National Meeting, Chicago, May.

30. NAUSS RM (1974) Parametric integer programming. PhD Dissertation, Graduate School of Management, University of California, Los Angeles, December. Available as Working Paper No. 226, Western Management Science Institute, UCLA.
31. NOONAN F (1973) Optimal investment planning for electric power generation. PhD Dissertation, Dep. of Industrial Engineering and Operations Research, University of Massachusetts, September.
32. PIPER CJ (1974) Implicit enumeration: a computational study. Working Paper No. 115, School of Business Administration, University of Western Ontario, London, Canada, May.
33. RADKE M (1975) Sensitivity Analysis in Discrete Optimization. PhD Dissertation, Graduate School of Management, UCLA, September.
34. ROODMAN GM (1972) Postoptimality analysis in zero-one programming by implicit enumeration. *Nav. Res. Logist. Q.* **19**(3), 435–447.
35. ROODMAN GM (1973) Postoptimality analysis in integer programming by implicit enumeration: the mixed integer case. Working Paper, Amos Tuck School of Business Administration, Dartmouth College, October.
36. ROSS GT and SOLAND RM (1975) A branch and bound algorithm for the generalized assignment problem. *Math. Progm.* **8**(1).
37. SOMMER D (1972) Computational experience with the Ophelie mixed integer code. Control Data Corp., Minneapolis, Minn. May.
38. THOMAS GS, JENNINGS JC and ABBOTT P (1973) A blending problem using integer programming on-line. Paper presented at the *VIII International Symposium on Mathematical Programming, Stanford University, August* (authors at Scicon, Ltd. London).
39. TOMLIN JA (1972) Survey of computational methods for solving large scale systems. Tech. Rep. 72–25, Dep. of Operations Research, Stanford University, October; reprinted in *Proceedings of the IEEE Conference on Decision and Control and 11th Symposium on Adaptive Processes, December.*
40. WHITE WW (1973) A status report on computing algorithms for mathematical programming. *Computg. Surv.* **5**(3), 135–166.
41. WILLIAMS AC (1973) Some modeling principles for M.I.P.'s. Paper presented at the *VIII International Symposium on Mathematical Programming, Stanford University, August* (author at Mobil Oil Corp. Princeton, N.J.).
42. WILLIAMS HP (1974) Experiments in the formulation of integer programming problems. *Math. Progm. Study* 2, December.
43. WINKLER C (1974) Basis factorization for block-angular linear programs: unified theory of partitioning and decomposition using the Simplex method. Tech. Rep. SOL 74–19, Systems Optimization Laboratory, Stanford University, December.

ADDRESS FOR CORRESPONDENCE: *Professor A Geoffrion, Western Mgmt Science Institute, University of California, Los Angeles, Calif 90024, USA.*