OMEGA, The Int. Jl of Mgmt Sci., Vol. 4, No. 5, 1976. Pergamon Press. Printed in Great Britain

Dynamic Strategies for Branch and Bound

FRED GLOVER

LEE TANGEDAHL

(Received July 1965; in revised form June 1976)

Some straightforward but unconventional strategies are proposed for implementing branch and bound methods. These include 'shrinking' the branch and bound tree and instituting 'branch reversals' by reference to the notion of the relative influence of particular branches in the current solution. An attractive feature of these strategies is their ease of implementation, and the fact that they free the solution process from its customary dependency on early branches created on the basis of inadequate information.

A GOOD many proposals for improved branch and bound methods develop rather intricate types of penalty calculations, problem relaxations, or 'multi-fork' branching alternatives (see, e.g. [1–8]). This note instead proposes a variety of straightforward strategies that are easily implemented, but that involve manipulating branches in non-standard ways. The following reflections and observations are offered informally as a possible spur to investigate by-passed alternatives.

One of the more vexing aspects of branch and bound is the fact that branching decisions must be based on very limited information about the likelihood that a particular branch will lead to an optimal solution. This limitation is especially pronounced at early stages of the branch and bound tree. At such stages, the small number of imposed branches does not yield sufficiently accurate bound calculations or other measures of branch desirability to provide very reliable estimates about which branch should preferably be imposed next. Yet the choices made at early stages can have a dominating influence on the efficiency of the branch and bound process. Even 'reasonable' choices can be extremely poor if they are not sufficiently influential to reduce the alternatives for other variables. Unless a current branch has the power to inhibit the range of remaining alternatives, the branch and bound process can degenerate into the disastrous approximation to total enumeration sometimes documented.

This research was partly supported by the Office of Naval Research, Contract N00014-76-C00383 with Decision Analysis and Research Institute, Inc., Boulder, Colorado, USA.

From this point of view, it seems worthwhile to consider a branching technique that has the ability to rid itself of certain types of uninfluential branches on the basis of more reliable information available at later stages. A chief component of this technique is to shrink the branch and bound tree by eliminating earlier branches that have been rendered 'conditionally superfluous' by subsequent decisions.

In particular, suppose that an integer variable x_j has been subjected to a branching constraint such as $x_j \leq k$ or $x_j \geq k$, where k is an integer. This branching constraint will be called 'currently uninfluential' if the updated LP objective-function coefficient for its slack variable ($s_j = k - x_j$ or $s_j = x_j - k$) is 0. To understand the motivation for this terminology, note that if x_j has been subjected to the restriction $x_j \leq 5$, and x_j receives a value of $4\frac{1}{8}$ in the current LP solution, then the slack variable $s_j = 5 - x_j$ is positive and hence basic. This implies the LP objective-function coefficient for s_j is 0. Consequently, the branch $x_j \leq 5$ qualifies as currently uninfluential—which is appropriate, because the branch does not actively restrict the value of x_j in the current solution. More generally, the stipulation that the objective-function coefficient for s_j is 0 is motivated by the fact that this implies the branching constraint does not affect the optimality of the current LP solution.

Regardless of the sequence in which branches are historically imposed, it is of course always possible to defer the decision about the sequence in which they are viewed as being imposed, for the purpose of implementing various tree search rules (such as the LIFO and 'best bound' rules). The only restriction to this deferred sequencing decision is that branches that historically precede a compulsory branch (that is, one whose alternative has been eliminated by examination or fathoming) cannot be re-sequenced to follow this branch. (This follows simply from the fact that compulsory branches rely on historically antecedent branches for their compulsory status.) The preceding notion of a currently uninfluential variable suggests the strategy not merely of re-sequencing historical decisions, but of actually discarding some of them, thereby collapsing the branch and bound tree.

For instance, it characteristically happens that some variables will automatically receive integer variables (e.g. 0) as a consequence of imposing branch restrictions on other variables. Yet from lack of reliable local information, variables that may be 'dependent' on others cannot be distinguished from their companions, and thus may be incorporated into the branching decisions. The ability to identify and release branches that subsequently are discovered to be uninfluential in the indicated sense can therefore be useful. The only restriction that must be observed is that a branch cannot be discarded if it precedes a compulsory branch—again for the obvious reason that the compulsory branch may depend on the antecedent branch for its compulsory status.

The same type of notion that prompts this tree shrinking strategy also suggests the possibility of ranking branches in terms of their current influence, as

measured by the magnitude of updated objective-function coefficients for slack variables associated with the branches. Such a ranking can be used to determine the sequential ordering of the branches in the branch and bound tree. For example, by analogy with the situation in which currently uninfluential branches may be discarded, more influential branches may be placed earlier in the tree and less influential branches later. (We shall call this a 'resequencing strategy.') This raises interesting issues, including a rationale for reversing as well as discarding branches. We briefly summarize these as follows.

(1) Since rankings will change depending on the current LP solution, the point at which a ranking imparts a sequence position to a given branch can make a difference in the structure of the branch and bound tree. This suggests keeping a history of influence measures for branches and biasing current influences in the direction of a weighted composite of historical influences.

(2) Influence measures communicate two competitive types of information. While it seems natural from one standpoint to place the more influential branches earlier in the tree, it seems equally natural from another standpoint to regard these branches as more likely to represent poor branching alternatives. In particular, the larger the objective-function coefficient attached to a given slack variable, the more 'expensive' the associated branch appears to be (since reversing the branch by allowing the slack to equal -1 will locally improve the objective function by the value of the objective-function coefficient).

These competitive information aspects can be made the basis of a 'branch reversal' strategy. For example, one may re-evaluate a seemingly expensive branch by setting its slack to -1 and re-optimizing with the dual method. Or more simply, one can use a single dual pivot update of the objective-function value. If setting the slack to -1 yields no infeasibility (or an infeasibility that is easily removed while maintaining an improved objective function value) then a branch reversal seems warranted. A reversal antecedent to a compulsory branch would destroy the compulsory status, but should at least be considered as a trial possibility upon reaching an integer feasible solution—since if such a reversal yields no infeasibilities it may immediately provide an improved candidate for an optimal mixed IP solution.

(3) The evaluation of branch reversal possibilities allows a refinement in the choice of a sequence for the current branches. Specifically, maintaining the view that influential branches should appear earlier in the sequence, it seems just as important for branches with relatively attractive alternatives to appear later, and an intelligent sequencing strategy will attempt to exploit both of these notions. Historical information concerning prior evaluation of branch alternatives can reasonably be used to supplement the sequencing decision.

All of these ideas can be implemented with relatively minor changes in current branch and bound procedures. They are proposed in this spirit, as simple

strategic notions whose time for study seems ripe—and that can be readily pursued by convenient modification of existing systems.

An example

To illustrate the operations of shrinking and resequencing, and to provide a clearer understanding of how they can affect the solution process, consider the following problem:

$$\begin{aligned} \text{Minimize } x_0 = \quad & 8x_1 + 8x_2 - 4x_3 \\ & 3x_1 \qquad - 2x_3 \geq 2 \\ & 2x_x - 2x_2 \qquad \geq 1 \\ & -8x_1 + 20x_2 \qquad \geq -1 \\ & x_1, x_2, x_3 \geq 0 \text{ and integer.} \end{aligned}$$

The following sequence of branch and bound steps, for one set of branching rules (whose form is unimportant for illustrating the shrinking and resequencing operations), identifies the values of the variables at each LP optimization by the vector $x = (x_1, x_2, x_3)$.

Step 1: Initial LP solution: $x = (0{\cdot}75, 0{\cdot}25, 0{\cdot}13)$
Branch: $x_1 \geq 1$.

Step 2: LP reoptimization: $x = (1, 0{\cdot}35, 0{\cdot}50)$
Branch: $x_3 \geq 1$.

Step 3: LP reoptimization: $x = (1{\cdot}33, 0{\cdot}48, 1)$
Shrink: drop $x_1 \geq 1$ Branch: $x_1 \geq 2$.

Step 4: LP reoptimization: $x = (2, 0{\cdot}75, 2)$
Shrink: Drop $x_3 \geq 1$ Branch $x_2 \geq 1$.

Step 5: LP reoptimization: $x = (2, 1, 2)$
Feasible integer solution: $x_0 = 16$
Resequence: current branch sequence $x_1 \geq 2, x_2 \geq 1$
new branch sequence $x_2 \geq 1, x_1 \geq 2$
Backtrack: impose $x_1 \leq 1$.

Step 6: LP reoptimization: no feasible solution
Backtrack: impose $x_2 \leq 0$.

Step 7: LP reoptimization: no feasible solution
Problem solution complete (no branches left).

In the preceding steps, note that the opportunity to shrink the tree occurred twice, the opportunity to resequence at backtracking occurred once, and these operations were implemented in each case. (In addition to the considerations

previously discussed, another useful criterion for resequencing, used here, is to position branches earlier in the tree for those variables whose values have covered the smallest range during the LP reoptimizations.) Using the same branch rules, the solution of the foregoing problem requires eleven steps (LP reoptimizations) if either the option to shrink or the option to resequence is bypassed, and requires fifteen steps if both options are bypassed.

Preliminary computational experience

To obtain a preliminary indication of the computational promise of dynamic manipulation strategies in branch and bound, we have programmed and tested the method on eight small problems, ranging from 17 to 25 integer variables and 9 to 21 constraints. In spite of their size, these problems have presented considerable difficulty to other solution methods (see [9, 10]). In fact, one of those problems, from DuPont, has more than 8 years of history as a tough problem, and the latest attempt to solve it with a commercial integer programming system required over 20 min on the IBM 360/65.

Our test results were extremely encouraging. The application of the shrinking and resequencing strategies resulted in cutting solution times for these problems by factors of approximately two to seven (21/12 sec and 71/11 sec) in the worst and best cases. The 'difficult' DuPont problem was solved in only 20 sec, using the CDC 6400. The question naturally arises whether the improvements from the shrinking and resequencing strategies were available only because the version of the B&B routine without them was poor. In fact, the opposite is the case. The B&B procedure in the absence of these strategies proved substantially more effective for solving the test problems than the methods cited in [9, 10]. This procedure succeeded in solving the DuPont problem, for example, in 87 sec.

More extensive tests on wider ranges of problems, employing branch reversal strategies as well as shrinking and resequencing strategies, are currently under way.

REFERENCES

1. ARMSTRONG RD and SINHA P (1974) Improved penalty calculations for a mixed integer branch-and-bound algorithm. *Mathl Prog.* **6**, 212–213.
2. BALAS E (1972) Integer programming and convex analysis: intersection cuts from outer polars. *Mathl Progr.* **2**(3),
3. BREU R and BURDET CA (1974) A subadditive approach to the group problem programming. *Mathl Progr.* **2**, 1–50.
4. FISHER ML, NORTHRUP WD and SHAPIRO JF (1974) Using duality to solve discrete optimization problems: theory and computational experience. Working Paper OR 030-74, Operations Research Center, MIT.
5. GEOFFRION AM (1974) Langrangean relaxation for integer programming. *Mathl progr.* **2**, 83–114.

6. GLOVER F (1973) Polyhydral annexation in mixed integer programming. MSRS 73-9, University of Colorado.
7. JEROSLOW RG (1974) Relaxations of integer programs. Management Science Research Report No. 347, Carnegie-Mellon University.
8. TOMLIN JA (1971) An improved branch and bound method for integer programming. *Ops Res.* **19,** 1070–1075.
9. WHITE CH (1976) Discrete blending (trouble in small packages). Paper presented at the Joint National ORSA/TIMS Meetings.
10. WOOLSEY RED (1973) Difficult integer programming problems. Colorado School Mines, Golden, working paper.

ADDRESS FOR CORRESPONDENCE: *Professor Fred Glover, Business Research Division, Graduate School of Business Administration, University of Colorado, Boulder, Colorado* 80302, *USA.*