# An evaluation of integer programming in forest production scheduling problems

**Article** · December 2015

CITATIONS
4

READS
50

**2 authors**, including:

B. Bruce Bare
University of Washington Seattle
**63** PUBLICATIONS **861** CITATIONS

SEE PROFILE

research bulletin No. 847
April 1969

# An Evaluation of Integer Programming In Forest Production Scheduling Problems

B. Bruce Bare and Eldon L. Norman
*Department of Forestry and Conservation*

ONE OF THE PRINCIPAL FUNCTIONS of a forest manager is the preparation of cutting schedules. These schedules, essential to efficient forest planning, transform the allowable cut decision into a working plan of action. Therefore, the forest manager usually attaches significant importance to their development. If the manager visualizes his scheduling problems in terms of either maximizing or minimizing an objective function subject to a set of constraints, it is possible for him to develop optimum schedules over a fixed planning period. This, however, requires familiarity with mathematical programming procedures.

This bulletin describes a potentially useful mathematical programming technique known as integer linear programming. While no new integer programming (IP) algorithm is developed, a discussion of several known IP algorithms is undertaken and one of the more recent ones is subsequently applied to forestry oriented problems.

Although computational problems have hampered the practical application of many mathematical programming techniques, the development

PURDUE UNIVERSITY
Agricultural Experiment Station, Lafayette, Indiana

of the simplex method virtually solved the computational problems associated with linear programming (LP). Partly because of this, foresters have used LP for solving problems dealing with the planning and control of forest production scheduling processes (Curtis 1962, Donnelly et al. 1963, Kidd et al. 1966, Leak 1964, Liittschwager and Tcheng 1967, Loucks 1964, Nautiyal and Pearse 1967, Norman and Curlin 1968, Theiler 1959, Wardle 1965). However, the optimal linear programming solution often specifies that fractional portions of a stand or compartment receive the impact of a management activity. In cases where forest stands or compartments are being managed as homogeneous units, non-integral solutions are impractical. Thus, foresters must turn to a procedure, such as integer programming, where all variables entering the optimal solution are restricted to integral values. The principal disadvantage of using IP is that an efficient algorithm for handling large problems has yet to be developed.

### Introduction to Integer Programming

This portion of the bulletin briefly reviews the major areas of thought which characterize the past and present attempts to derive an efficient integer programming algorithm. The discussion of the following methods will be brief and intuitive in nature. A more detailed discussion may be found in Balinski (1965).

### Cutting Hyperplanes

Acording to Young (1964), two distinct steps characterize any cutting plane algorithm. The first involves the generation of a basic trial solution to the linear programming problem using a standard linear programming algorithm. The second involves the generation of an additional constraint on the set of feasible solutions (called a cutting hyperplane) if the LP solution does not produce an optimum integer solution. Cutting plane algorithms are distinguished by the characteristics of the basic solutions they utilize and the method used to generate cuts.

Initial efforts in this field by Dantzig et al. (1954), Dantzig (1959), Markowitz and Manne (1957), and Gomory (1958) shared a common method of obtaining basic and trial solutions but differed in their method of cut generation. Gomory's work is distinguished because he: (a) developed a proof of finiteness and (b) stimulated the development of other IP algorithms which were based on his work. His original algorithm requires an optimum LP solution as an initial trial solution. If this optimal solution does not satisfy the required integer property, the algorithm generates a hyperplane from a row with a fractional solution variable. The resulting tableau which is dual, but not primal, feasible is then re-optimized using the dual simplex. This cycle is repeated until an integer optimal solution is found, or until all coefficients in the generating row are non-negative—which implies infeasibility.

This algorithm has since been superseded by other types of cutting plane algorithms which employ a common method of generating cuts, but differ in the character of the basic solutions they generate. The Gomory (1960) all-integer integer algorithm which also uses the dual simplex to locate trial and basic solutions, requires that all of the $a_{ij}$ in the initial simplex tableau be integer. This algorithm generates a sequence of basic solutions which are integral and optimal (i.e., dual feasible) but not primal feasible before the final iteration. The basic idea involves the generation of a hyperplane from a row with a negative solution variable in such a way that the pivot element in the generated row is $-1$. The positive integer $\lambda$, by which the generating row is divided to form the hyperplane, is chosen so that dual feasibility is maintained.

### Implicit Enumeration

Implicit enumeration algorithms may be defined as methods which generate information, as the enumeration of possible solutions proceeds, which permit the elimination of large numbers of solutions from further consideration. For this reason they are also known as branch and exclude algorithms. One of the earliest algorithms in this area was that developed by Balas (1965).

The Balas additive algorithm, and implicit enumeration methods in general, have several advantages over the previously discussed cutting hyperplane approaches. Several of these advantages are:

a. addition is the only arithmetic operation required (hence no system of simultaneous equations need be solved as in LP)
b. the rate of implicit enumeration can be measured, thus allowing for informed stopping rules
c. only minor modifications are necessary for the solution of nonlinear objective functions.

In the solution of the two scheduling problems presented below, the Geoffrion (1967) algorithm (a reformulation of the Balas procedure) is used. Therefore, the following intuitive discussion of implicit enumeration is only directed at the Geoffrion algorithm.

Geoffrion's implicit enumeration approach utilizes the Balas (1965) algorithm with the modification of the backtracking scheme proposed by Glover (1965). His algorithm states that any bounded integer programming problem can be written as

$$\text{Min} \quad cx$$

subject to

$$Ax + b \geqslant \bar{0}$$
$$x_j = 0 \text{ or } 1$$

where

c is an n-vector, b and $\bar{0}$ are m-vectors, A is an mxn matrix, and x is a binary vector to be chosen.

The reader may consult Geoffrion (1967) or Freeman (1965) if interested in a brief discussion concerning conversion of non-binary integer variables with an upper bound to binary representation.

The basic idea of this and other implicit enumeration techniques is to explicitly enumerate a subset of the $2^n$ possible solutions in a non-redundant fashion. However, there is no guarantee that the number of solutions explicitly enumerated will not approach $2^n$. The key to an efficient implicit enumeration technique lies in the scheme which permits large groups of possible solutions to be excluded (implicitly enumerated) from further consideration.

Geoffrion's algorithm involves the generation of a sequence of partial solutions, and the subsequent completion of each. A partial solution is defined as an assignment of binary values to a subset of the n variables, with any variables not assigned a value being designated as free. A completion of a partial solution is defined as an assignment of binary values to the free variables not included in the partial solution. As feasible solutions are discovered, that which minimizes cx is retained as the incumbent solution. For a given partial solution, the best feasible completion is computed. If the resulting value of the objective function is better than the value of the incumbent solution, then it replaces the latter in memory. Otherwise, the incumbent solution remains unchanged. In either case, the above process is defined as fathoming a partial solution. As the fathoming process proceeds, it is possible to dedetermine whether the subsequent completion of a particular partial solution will produce a better value of the objective function than will the incumbent solution. If no feasible solution exists with a better value, then the remaining possible completions are said to be implicitly enumerated and the partial solution is fathomed. This process is repeated until the optimum integer solution is found.

Another facet of the implicit enumeration approach involves the concept of a filter on the possible branches of the tree search. Constraints are generated using the solution of the continuous dual of the primal problem. These constraints, called surrogate constraints, are created by forming linear combinations of the primal constraint set and the optimum dual continuous variables. The original filter concept is due to Benders (1962). However, since Balas had not published his original implicit search article at that time, Benders was unable to link the filter with Balas' implicit search technique. Balas (1967), Glover (1965), and Geoffrion (1968) have since combined the filter concept with implicit search techniques to produce new algorithms.

Another recently developed approach involves the use of probability theory in an implicit search (Graves and Whinston 1967). Its strength lies in a more powerful means of determining the potential future effect, with respect to the problem constaints, of bringing an additional variable into solution.

### Two Sample Scheduling Problems

The first sample scheduling problem concerns a forest which has been subdivided into m compartments with each compartment to receive a single harvest cutting over the next n years. Let $v_{ij}$ equal the total cubic foot volume received from compartment i if harvested in year j, and let $x_{ij}$ be a binary variable which equals 1 if compartment i is harvested in year j and 0 otherwise. Hence, we wish to:

$$\text{Max} \sum_{j=1}^{n} \sum_{i=1}^{m} v_{ij} \quad x_{ij} \qquad (1)$$

In addition, there are minimum and maximum restrictions on the number of acres that may be harvested any one year. Using the notation of Liittschwager and Tcheng (1967), let $a_i$ equal the number of acres in compartment i, and $b_j^{max}$ and $b_j^{min}$ equal the maximum and minimum number of acres which may be harvested each year. Thus, the following inequalities are obtained:

$$\sum_{i=1}^{m} a_i \quad x_{ij} \leqslant b_j^{max} \qquad (2)$$
$$j = 1, 2, \ldots . n$$
$$\sum_{i=1}^{m} a_i \quad x_{ij} \geqslant b_j^{min} \qquad (3)$$

The problem may be further simplified by setting $b_j^{max}$ and $b_j^{min}$ equal to the acreage of the largest

and smallest compartments respectively. Thus, $b_j^{max}$ and $b_j^{min}$ will be constant over all n years in the planning horizon.

Two additional constraints are needed to complete the mathematical formulation. As stated above, each compartment must be harvested once and only once during the planning horizon. Writing this as an equality we obtain:

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, 2, \ldots, m, \; x_{ij} = 0 \text{ or } 1 \qquad (4)$$

This problem formulation involves (mxn) binary variables and 2n + m constraints. Thus, a simple scheduling problem involving the harvesting of 5 compartments over a planning horizon of 5 cutting periods, with only one compartment being harvested per cutting period, involves 25 binary variables and 15 constraints. Table 1 contains the volumes and acreages of the five compartments for the total planning horizon.* Although this problem involves only 15 constraints, the algorithm (Geoffrion's) used to solve the problem does not permit the use of equality constraints. Thus, the following two constraints (instead of one) are required for each equality constraint:

$$\sum_{j=1}^{n} x_{ij} \leqslant 1 \text{ and } -\sum_{j=1}^{n} x_{ij} \geqslant -1 \qquad (5)$$

or

$$\sum_{j=1}^{n} x_{ij} \geqslant 1 \text{ and } -\sum_{j=1}^{n} x_{ij} \leqslant -1 \qquad (6)$$

Therefore, the actual size of the constraint matrix for this problem is (25 x 20). However, both Balas (1965) and Reiter and Rice (1966) state that the efficiency of the integer programming algorithm seems to increase as the number of constraints increase.

A computer program version of Geoffrion's algorithm written for the CDC 6500 was used to solve this problem in 32 seconds. The optimal value of the objective function was 2467M cu. ft., and is associated with the optimal harvest schedule shown in Table 2. It is interesting to note that the first feasible solution, found in 3½ seconds, provided an objective function value which was 98.6% of the optimal value.

* The compartment acreages used in this sample problem prohibit more than one compartment from being harvested in any given cutting period. Therefore, constraint equations (2) and (3) become redundant, and the problem can be solved by using a transportation algorithm. However, if different compartment acreages are chosen it is likely that it will not be possible to formulate this as a transportation problem. Thus, for increased flexibility, sample problem one was not considered as a transportation problem.

**Table 1. Data for sample problem one**

| Compt. No. | Acreage (Acres) | Cubic foot volume per compt. (thousand cu. ft.) | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| 1 | 481 | 400 | 480 | 491 | 500 | 501 |
| 2 | 580 | 500 | 510 | 510 | 490 | 480 |
| 3 | 299 | 300 | 340 | 370 | 380 | 385 |
| 4 | 360 | 461 | 450 | 440 | 420 | 400 |
| 5 | 295 | 564 | 581 | 600 | 620 | 620 |

It was not too surprising to obtain the final optimum solution in such a short time since both Petersen (1967) and Freeman (1965) report that the original Bales algorithm seems to work well for problems involving up to 30 variables. In addition, the problem was fairly well constrained, which as stated above seems to increase computational efficiency.

Next we increased the number of compartments and the number of cutting periods to ten. This rather simple scheduling problem involved 100 binary variables and 30 constraints. After 30 minutes of CDC 6500 central processor time no optimal solution was obtained. Even more alarming, the initial feasible solution (obtained in 38 seconds) was within 9% of the value obtained after an additional 1762 seconds of computing. In addition, the last solution obtained was only within 22% of the known optimal feasible solution.

Although not shown in Table 2, it was observed that the objective function was quite insensitive to different harvesting schedules. Thus, a substantial reordering of the harvest schedule only slightly affected the value of the objective function. This may indicate that an intuitive subjective schedule is not too far from the optimum.

The second sample scheduling problem concerns a forest which has been subdivided into m stands, with each stand being subjected to n possible harvesting alternatives. Let $r_{ij}$ equal the residual value of stand i if harvesting alternative j is selected, and let $x_{ij}$ equal 1 if stand i is subjected

**Table 2. Optimal schedule for sample problem one.**

| Cutting Period | Compartment to Harvest |
|---|---|
| 1 | 4 |
| 2 | 2 |
| 3 | 1 |
| 4 | 5 |
| 5 | 3 |

to harvesting alternative j, and 0 otherwise. Hence, we wish to:

$$\text{Max} \sum_{i=1}^{m} \sum_{j=1}^{n} r_{ij} \quad x_{ij} \qquad (7)$$

In addition, there is a maximum volume which may be removed during the harvesting operation. Let $v_{ij}$ equal the volume available for harvesting from stand i if harvesting alternative j is selected. To maintain an adequate growing stock, a harvest volume of no more than Y bd.ft. may be removed. This gives rise to to the following inequality:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} v_{ij} \quad x_{ij} \leqslant Y \qquad (8)$$

There is an additional constraint concerning the stumpage value of the stands to be harvested. Let $s_{ij}$ equal the stumpage value of stand i if harvested by alternative j. The stumpage value for the entire harvesting operation must be greater than or equal to Z dollars.

Writing this an an inequality we obtain:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} s_{ij} \quad x_{ij} \geqslant Z \qquad (9)$$

Two additional constraints are needed to complete the problem formulation. Each stand must be completely harvested if any cutting is done in the stand, and $x_{ji}$ must equal either 1 or 0, i.e., equation set (4).

Assume that we have 10 stands with 9 harvesting alternatives per stand. This gives rise to a problem involving 90 binary variables and 12 constraint equations. For reasons noted in equations (5) and (6), we must increase the number of constraints to 22. Further, we have a maximum harvesting volume of 850,000 bd.ft. and we must sell a minimum of $10,000 worth of stumpage. For this problem we know the optimum LP solution, but not the optimum IP solution.

Using the same computer program version of Geoffrion's algorithm referred to above, an optimum solution was not found after 20 minutes of computing. However, the incumbent solution after 12.5 seconds was only 3.66% less than the optimum linear programming solution. It is not known whether this is the optimum integer solution since all solutions were not implicitly enumerated in the 20-minute time limit set for the run. Because the optimum integer solution must be less than or equal to the optimum LP solution, this IP solution may be optimal.

Table 3, which contains the LP and IP harvesting schedules for this problem, illustrates several interesting aspects of the problem. In comparing the LP and Geoffrion solutions, it again appears that the objective function is not particularly sensitive to the variables in the solution set since the Geoffrion schedule produced a similar value of the objective function with only three of the ten selected variables in common. This indicates that a large number of near optimum solution sets probably exist.

**Table 3. Solution comparisons for sample problem two**

| Stand | Optimum LP Schedule Alternative Number | Residual Value | Geoffrion Schedule Alternative Number | Residual Value |
|---|---|---|---|---|
| 1 | 4 | 5,625 | 1 | 6,950 |
| 2 | 3 | 2,899 | 7 | 1,321 |
| 3 | 1 | 4,798 | 3 | 4,661 |
| 4 | 6 | 4,324 | 6 | 4,324 |
| 5 | 3-51%<br>4-49% | 599 | 5 | 0 |
| 6 | 4 | 3,155 | 6 | 2,485 |
| 7 | 4 | 8,533 | 3 | 9,737 |
| 8 | 3 | 4,177 | 3 | 4,177 |
| 9 | 2 | 2,239 | 7 | 1,415 |
| 10 | 3 | 2,398 | 3 | 2,398 |
| Max | $\Sigma_i \Sigma_j r_{ij} x_{ij}$ | $38,983 | | $37,468 |

## Conclusions

To completely evaluate the performance of some of the proposed integer programming algorithms it would be necessary to solve many different types of scheduling problems. Moreover, the two specific harvest scheduling problems discussed in this bulletin should be solved by some of the other integer programming algorithms. However, even this approach would not necessarily permit one to draw general conclusions concerning the efficiency or usefulness of a specific algorithm for solving the harvest scheduling problem. The reasons for this are: (a) one algorithm may be more efficient for solving one specific formulation while a second algorithm may be more efficient for solving another formulation, and (b) the efficiency of the proposed algorithms is directly related to the size of the problem. However, as Freeman (1965) states, "as with any enumeration procedure computational experience is the best indicator of its worthiness." Thus, additional computational experience may be warranted for the purpose of determining the most efficient algorithm to use for a specific formulation of the harvest scheduling problem.

The implication is that there is no "best" integer programming algorithm even for problems of a similar structure. Perhaps the future will reveal a method similar to the simplex procedure which will have wide applicability. However, the trend seems to be to develop implicit enumeration techniques which work very efficiently for problems of a certain structure, but very inefficiently for problems of a different structure. In addition, computational times vary almost exponentially with the size of the problem.

Two encouraging items which we observed were: (a) an initial feasible integer solution was obtained in a very short time, and (b) the value of the objective function associated with the initial solution was often quite close to the value of the objective function associated with the optimum solution. However, the problem containing 100 binary variables did not follow this pattern because the initial and final (not optimal) objective function values were not even close to the optimum value.

A major limitation of the implicit enumeration approach is due to the fact that the optimum is unknown, even though it may have been found, before all solutions have been implicitly enumerated. Although it is difficult to generalize on the basis of two tests, this suggests that one could merge an implicit enumeration algorithm with another IP solution procedure to obtain a more efficient computational algorithm. This observation was also noticed by Freeman (1965).

Another observation is that integer programming tends to place more emphasis on careful problem formulation. Hence, to reduce the number of decision variables, attempts should be made to delete alternatives which are of minor importance to the problem. Also, as previously noted, the Geoffrion algorithm increases the size of the problem when equality constraints are used. In addition, and more important computationally, is the fact that Geoffrion's algorithm is designed for minimization problems. Therefore, when maximization problems of the type discussed in this bulletin are encountered, a transformation ($x_i = 1 - x_i$) is required. This increases the number of variables in a partial solution, and thus slows down the computational efficiency of the algorithm. Methods for alleviating this problem are currently under study.

It is apparent that, at present, the relatively large nature of practical forest scheduling problems and the efficiency of existing integer programming algorithms are not compatible. However, the characteristics of the general structure of many forest scheduling problems indicate that the problem is amenable to solution. Since the coefficient matrix is tied over all variables by relatively few constraints, and since the matrix tends to be uniform and independent in blocks across the variables, it lends itself to decomposition. The potential of this procedure should be evaluated.

From the preceding description of the constraint matrix, it follows that the density is generally low. Simplex algorithms have taken advantage of this characteristic, but as yet implicit enumeration methods have not. We concur with Lemke-Spielberg (1967) who feel that the area of implicit enumeration is in the infant stage and eventually will become much more successful as more sophisticated search methods are developed. In addition, the newer filter methods and the probability approach should be evaluated through computational experience.

## Literature Cited


Balas,E. 1965. An Additive Algorithm for Solving Linear Programs with Zero-One Variables. J. Operations Res. 13(4):517-549.

Balas, E. 1967. Discrete Programming by the Filter Method. J. Operations Res. 15(5):915-957.

Balinski, M. 1965. Integer Programming: Methods, Uses, Computation. Mgt. Sci. 12(3):253-313.

Benders, J. F. 1962. Partioning Procedures for Solving Mixed-Variables Programming Problems. Numerische Mathematik 4:238-252.

Curtis, F. 1962. Linear Programming the Management of a Forest Property. J. For. 60:611-616.

Dantzig, G. B., D. R. Fulkerson, and S. Johnson. 1954. Solution of a Large-Scale Traveling-Salesman Problem. J. Operations Res. 2:393-410.

Dantzig, G. B. 1959. Note on Solving Linear Programs in Integers Naval Res. Logistics Qrtly 6:75-76.

Donnelly, R. H., R. W. Gardner, and H. R. Hamilton, 1963. Integrating Woodlands Activities by Mathematical Programming. Battelle Memorial Institute (for American Pulpwood Association).

Freeman, R. 1965. Computational Experience with the Balas Integer Programming Algorithm. RAND Corp. P-3241.

Geoffrion, A. 1967. Integer Programming by Implicit Enumeration and Balas' Method. SIAM Review. 9(2):178-190.

Geoffrion, A. 1968. An Improved Implicit Enumeration Approach for Integer Programming. Western Mgt. Sci. Inst. Working Paper No. 137.

Glover, F. 1965. A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem. J. Operations Res. 13(6):879-919.

Gomory, R. E. 1958. An Algorithm for Integer Solutions to Linear Programs. Princeton-IBM Math. Res. Project Tech. Report 1.

Gomory, R. E. 1960. All-Integer Integer Programming Algorithm. IBM Res. Report RC-189.

Graves, G. W. and A. B. Whinston. 1967. A New Approach to Discrete Mathematical Programming. Grad. School Ind. Adm. Purdue Univ.

Kidd, W., E. Thompson and P. Hoepner. 1966. Forest Regulation by Linear Programming. J. For. 64(9): 611-613.

Leak, W. B. 1964. Estimating Maximum Allowable Timber Yields by Linear Programming. Northeast For. Expt. Sta. Res. Paper 17. 9 pp.

Lemke, C. E. and K. Spielberg. 1967. Direct Search Algorithms for Zero-One Mixed-Integer Programming. J. Operations Res. 15(5):892-914.

Liittschwager, J. and T. Tcheng. 1967. Solution of a Large-Scale Forest Scheduling Problem by Linear Programming Decomposition. J. For. 65(9):644-646.

Loucks, D. 1964. Development of an Optimal Program for Sustained Yield Management. J. For. 62(7):485-490.

Markowitz, H. M. and A. S. Manne. 1957. On the Solution of Discrete Programming Problems. Econometrica 25:84-110.

Nautiyal, J. C. and P. H. Pearse. 1967. Optimizing the Conversion to Sustained Yield—A Programming Solution. For. Sci. 13:131-139.

Norman, E. L. and J. W. Curlin. 1968. A Linear Programming Model for Forest Production Control. Oak Ridge National Laboratory Report No. 4349.

Petersen, C. 1967. Computational Experience with Variants of the Balas Algorithm Applied to the Selection of R & D Projects. Mgt. Sci. 13(9):736-750.

Reiter, S. and D. Rice. 1966. Discrete Optimizing Solution Procedures for Linear and Nonlinear Integer Programming Problems. Mgt. Sci. 12(11):829-850.

Theiler, T. 1959. Linear Programming and Optimal Cutting Practices. Paper Industry. 41(6).

Wardle, P. 1965. Forest Management and Operational Research, A Linear Programming Study. Mgt. Sci. 11(10):260-270.

Young, R. D. 1964. A Primal (All Integer) Integer Programming Algorithm. Working Paper 52. Grad. School Bus. Stanford Univ.