Integer Programming Methods for a Vessel Scheduling Problem

LEIF H. APPELGREN

Salén Shipping Companies, Stockholm, Sweden

In a previous paper, APPELGREN *(1969), a decomposition algorithm for a class of vessel scheduling problems was presented. In some problems, the algorithm gives fractional solutions that cannot be interpreted as feasible schedules. This paper treats two integer programming methods that can be used to resolve these cases. The cutting plane method that was first tested was abandoned because it was not able to solve all the test problems. The second method is a branch-and-bound algorithm, where the branching is performed on one of the 'essential' fractional variables and where the bounds are obtained by the decomposition algorithm. All fractional problems that have been found by simulation or in regular use of the algorithm have been solved, mostly with one branching only. There are fundamental difficulties in combining these integer programming methods with the* DANTZIG-WOLFE *decomposition, since the constraints generated in the master program have to be taken into account in the solution of the subprograms. The success in this case is due to the simple structure of the master LP problem.*

The economic background of the problem is described thoroughly in a previous paper, APPELGREN.[1] Therefore, this section is devoted mainly to the mathematical structure. The problem is to assign an optimal sequence of cargoes to each vessel in a given fleet during a given period. Each cargo is characterized by its revenue, type, size, loading and discharging ports, and loading and discharging dates. The loading date may vary in an interval, the 'loading date interval,' in which case the discharging date is varied accordingly. Each vessel is characterized by size, capability to

carry different types of cargo, speed, initial position, open date and 'time value.' The time value for a vessel is the expected daily marginal revenue from cargoes that are not known when the schedule is made. This value is collected for the time between the discharge of the last cargo in the cargo sequence and the end of the planning period.

A feasible cargo sequence for a vessel is a sequence of cargoes where the vessel has the size and capability to carry each of the cargoes and where the time between discharging and loading of consecutive cargoes is sufficient for the ballast trip between the corresponding ports.

If the problem contains one vessel only, it can be visualized as a network flow problem with one node for each loading date alternative of each cargo that can be carried by the vessel. Two nodes are connected by an arc if the time for the ballast trip is sufficient. It is assumed that the loading date interval cannot be so large that the same cargo can be shipped twice by one vessel. The nodes are given a value equal to the revenue of the cargo, and the arcs are given the cost or revenue of the ballast trip and the idle period. The problem is obviously a shortest route problem that can be solved easily by a dynamic programming algorithm. In a multivessel problem, different networks are obtained for different vessels because of differences in speed, size, initial position, etc. The networks are interdependent via capacity constraints on the nodes, which state that each cargo may only be carried once.

The algorithm described in the previous paper uses DANTZIG-WOLFE decomposition in order to separate the problem into independent network problems for each vessel. The master program takes care of the cargo constraints, which means that the master matrix consists of one row for each cargo and one convexity row for each vessel. Each column represents a cargo sequence for one vessel, with unit elements in the corresponding vessel row and in the rows corresponding to the cargoes in the cargo sequence. The right-hand side consists normally of unit elements, and the objective row contains the net revenue for each cargo sequence in the matrix. If several cargoes have identical data, they can be represented by the same cargo row, where the right-hand side is equal to the number of identical cargoes.

The fact that a linear programming matrix and right-hand side consist of unit elements only is no guarantee for integer solutions, which was confirmed by the experiments reported in the previous paper. In the following table a simple example is given that shows that a two-vessel, two-cargo problem can have a fractional solution. The optimal basis consists of columns 1, 4, 6, and 7 and the basic variables are all equal to 0.5.

Revenue	40	15	0	0	25	15	0	R.H.S.
Vessel row 1	1	1	1	1	0	0	0	1
Vessel row 2	0	0	0	0	1	1	1	1
Cargo row 1	0	1	0	1	0	1	0	1
Cargo row 2	0	0	1	1	0	0	1	1

Simulation of a large number of problems with 10–15 vessels and 15–25 cargoes indicated that the frequency of fractional solutions was 1–2 per cent. It could then be expected that integer programming methods should be quite successful in this problem for two reasons. First, the frequency of fractional solutions is small, which means that quite small changes in a problem with a fractional solution will probably give a new problem having an integer solution. Second, the structure of the problem is very simple, with zero-one elements in the LP matrix and the right-hand side. On the other hand, special problems arise because the LP problem is a master program in a decomposition algorithm, which means that the restrictions and the changes in the master program, which are created by an integer programming algorithm, also have to be implemented in the potential columns that are not presented in the master program, i.e., the extreme points of the subprograms.

In the rest of this paper, two integer programming methods are presented. The first method that was tried was the cutting plane algorithm presented in section 2. The generation of the new constraints, the 'cutting planes,' is based on simplified assumptions concerning the structure of the basic matrix in order to make the implementation in the subprograms simple and powerful. This means, however, that the algorithm cannot solve problems with a more complicated structure. Therefore, this method was abandoned when the second method proved to be more efficient.

The second method is the branch-and bound algorithm presented in section 1. The algorithm has up to now solved all problems where it has been applied, and the number of nodes required has been very moderate, ranging from 2 to 15.

1. THE BRANCH-AND-BOUND ALGORITHM

1.1 The Branches and the Bounds

For an introduction and a survey of the branch-and-bound technique, the reader is referred to LAWLER AND WOOD.[4] The method is based on the fact that the feasible solutions to a discrete optimization problem can be represented by the end points of the branches in a tree. The amount of search in such a tree is reduced by the use of upper bounds (in a maximiza-

tion problem) along the branches, which makes it possible permanently to exclude certain parts of the tree from the search.

A primitive way to apply the branch-and-bound technique to a scheduling problem is to let the first branching determine the first cargo for vessel 1, etc. The bounds can be obtained either by use of the LP model, which gives upper bounds because it optimizes without the constraint that the solution must be integer, or by simpler means. Experience tells that such a method would probably be too costly because the tree would become very large. The approach used here is to apply the tree-search technique only when it is necessary, i.e., when the decomposition algorithm gives a fractional solution. The branching is made with the objective of maximizing the probability that the solutions of the revised LP problems, whose values are used as the bounds in the approach, become integer.

Thus we select one column from the LP matrix whose corresponding variable is fractional. We study the two mutually excluding alternatives that this variable is set to zero and one, respectively. In both cases, the current fractional solution is prohibited, which should increase the probability of integer solutions in the two restricted problems. If an integer solution is obtained, this is the best feasible solution that can be obtained in this part of the tree, thus no more branching is needed from this node. The tree search goes on from the fractional node with the highest value until the best integer solution has a value that is not exceeded by any fractional bound. A hypothetical tree is shown in Fig. 1. This method is actually a straightforward application of the algorithm described by LAND AND DOIG.[3] The same approach has been used by LEVIN[5] on airline scheduling problems with the same positive experience that the solution trees terminate quite fast.

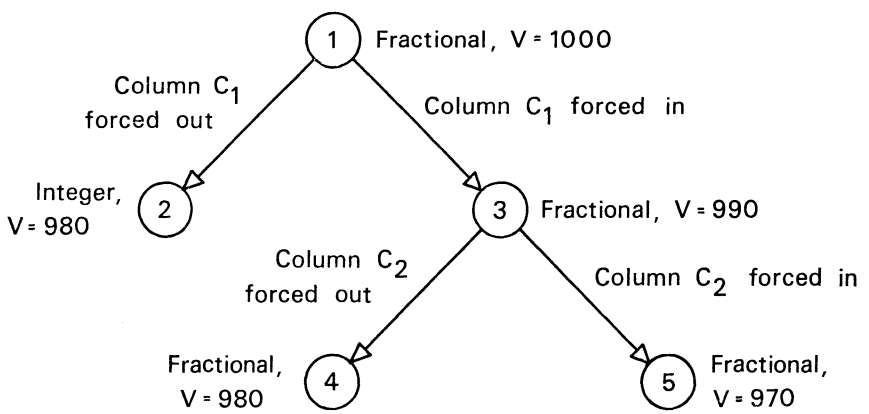

Fig. 1. A hypothetical branch-and-bound solution.

The selection of a critical column is described in the next section, and the implementation of the zero/one restrictions on such a critical variable is described in section 1.3. Finally, the results obtained with this method are reported in sec. 1.4.

1.2 Selection of a Critical Column

The basic variables of a fractional solution to the master program can be integer or fractional. The corresponding sets of columns are denoted C1 and CF. The rows of the matrix can be partitioned into two sets RF and RI, where the set RF, the set of fractional rows, is defined as the set of rows where the fractional columns have nonzero elements. The set RI consists of the remaining rows. After permutation of rows and columns, the matrix can be partitioned into four submatrices,

$$\begin{array}{c} \\ RF \\ RI \end{array} \begin{array}{c} \begin{array}{cc} CF & CI \end{array} \\ \begin{bmatrix} F & G \\ 0 & H \end{bmatrix} \end{array}.$$

We want to select one column from the set CF such that the new LP solutions obtained when the corresponding variable is forced to zero or one become integer with great probability. Therefore we want to exclude columns whose variables are fractional as a consequence of other variables being fractional. This is true for all columns in CF that have only one nonzero element, because the value of such a variable is determined by the other variables having nonzero elements in the same row. Next, we can delete the rows where those nonzero elements were located, because those rows cannot cause the fractional solution as they are only used for the determination of the single-element variables. In the reduced matrix, we can delete all the columns that only have one nonzero element, etc. This process is repeated until all remaining columns (the set CF_1) have at least two nonzero elements in the remaining set of rows RF_1. By this process we partition the matrix F into four parts,

$$\begin{array}{c} \\ RF_1 \\ \\ \end{array} \begin{array}{c} \begin{array}{cc} CF_1 & \end{array} \\ \begin{bmatrix} F_1 & 0 \\ F_2 & F_3 \end{bmatrix} \end{array}.$$

The matrix F_3 contains the nonzero elements of the deleted columns and the matrix F_2 contains the remaining elements of the deleted rows. The matrix to the right of F_1 is zero because if it contained a nonzero a_{ij}, row i would have been deleted when column j was deleted.

The basic equation system $Bx = b$ has a unique solution by definition.

From this fact follows that the system $Fx_f = b_f$ has a unique solution, which implies that the number of rows in the matrix F is greater than or equal to the number of columns in CF.

From the construction of the matrix F_1 follows that at least one column is deleted each time a row is deleted, thus the matrix F_3 must contain at least as many columns as rows. If there are more columns than rows, however, the vector x_3 of variables corresponding to these columns will not be determined uniquely, thus F_3 is a square matrix. If F_3 is singular, the vector x_3 will not be unique, thus F_3 is a square, nonsingular matrix. Thus x_3 can be expressed in x_1,

$$x_3 = -F_3^{-1}F_2x_1,$$

which means that the cause of fractional solutions is traced to the system $F_1x_1 = b_1$. The number of rows in F_1 is greater than or equal to the number of columns and the rank is equal to the number of columns. It is obvious that one of the columns CF_1 should be selected as the 'critical' column.

Some additional rules are used in the selection of the critical column. The first rule involves the computation of determinants of submatrices obtained from F_1. We denote the number of rows and columns in F_1 by m and n, respectively. The vector x_1 corresponding to the columns can be determined from any nonsingular set of n rows in F_1. Thus it follows from Cramer's rule that the denominator in the fractional variable values can never exceed the absolute value of the determinant of any such square submatrix.

When the critical variable has been selected and set to zero or one, the critical column can never occur again in the matrix F_1. It is thus reasonable to delete one column at a time and compute all nonsingular $(n - 1) \times (n - 1)$ determinants that can be obtained from the resulting $m \times (n - 1)$ matrix. The minimum absolute value of these determinants tells us how fractional a solution can be, assuming that no other 'bad' columns are generated, of course. Therefore, we ought to select the column that has the smallest determinant. As the number of different determinants can become quite large when $m > n$, we only compute one arbitrary nonzero determinant for each column and select the column whose determinant has the smallest absolute value.

Most of the time we have also used the rule that the selected column must have at least two nonzero elements in cargo rows in F_1. The reason for this is that if F_1 consists entirely of columns with one element in a vessel row and one element in a cargo row, F_1 is the basis of an assignment problem that always has an integer solution. Therefore it ought to be more efficient to select a column with two or more cargoes in F_1. We have noticed that the tree sometimes becomes much larger when this rule is abandoned.

1.3 Implementation of the Restrictions in the Decomposition Algorithm

In the branch-and-bound algorithm, we select a column in the LP problem and set the corresponding variable to either zero or one, after which the decomposition algorithm is applied again. This would be a trivial task in an ordinary LP problem but is quite complicated in a column generation algorithm, since a prohibited column can be generated again from the subprograms if no restrictions are imposed in order to prevent this. The opposite case where a certain column is forced into the solution can be handled easily by deletion of the corresponding vessel and cargoes from the problem. The value of this cargo sequence can then be added to the value of the reduced LP problem.

Thus the difficult task is to prevent a specific cargo sequence for a specific vessel without restricting any other cargo sequences. This can be done by means of a dynamic programming algorithm that also generates the second best solution. If the prohibited cargo sequence is the best one, the next best sequence is used instead. As several columns may be prohibited at some stage in the tree search, an 'nth best' algorithm would be required.

The approach presented above has not been used for two reasons. First, an ordinary dynamic programming algorithm is faster than an nth best algorithm. Second, it seems better to confine the restrictions to the elements in a column that are included in the essential matrix F_1. As an example, we assume that we have selected a column with nonzero elements in vessel row 1 and in cargo rows 1 and 10, i.e., the column implies that vessel 1 carries cargo 1 and 10. We assume also that vessel row 1 and cargo row 1 belong to the essential matrix F_1. If this sequence is prohibited, a similar fractional solution can occur again with cargo 10 replaced by a similar cargo 11. Thus it seems better to prohibit all columns where vessel 1 carries cargo 1. This can be done easily in the dynamic programming algorithm just by deleting cargo 1 when the subprogram for vessel 1 is solved. Along the opposite branch, cargo 1 shall be forced to be carried by vessel 1, which can be done by an increase in the revenue of cargo 1 for vessel 1 only.

In section 1.2 above, it was assumed that a column with nonzero elements in two cargo rows is more critical for the generation of fractional solutions. Such a two-cargo combination should, with the same reasoning as above, be prohibited for all vessels, since if it is prohibited only for the vessel used in the current LP solution, the combination may reppear on another vessel. A combination of two cargoes can be controlled easily in the dynamic programming algorithm, provided that the two cargoes always are consecutive, by variation of the revenue of the ballast trip between the cargoes. If it is possible to carry another cargo in the time between the two selected cargoes, this simple pricing operation does not work. One simple way to

solve the problem in that case is to run the dynamic programming algorithm twice, the first time without the first cargo and the second time without the second cargo. The best value from those two runs is then the best solution to the subprogram. This may work well in the first branch of the tree, but on the kth level in the tree, where k cargo combinations are involved, the subprograms have to be solved 2^k times, which might become prohibitive.

The algorithm was originally based on cargo combinations with two consecutive cargoes. Thus, the two selection rules stated in the previous section were actually augmented by a third rule, stating that the cargo combination must consist of two consecutive cargoes. This means that there might exist cases when no branching column can be selected, but this has not happened in any of the solved problems. This risk was taken because it was considered preferable to branch on a column with at least two cargoes in the essential matrix and to use this restriction on all vessels instead of only the vessel used in the original solution.

The consecutive cargo combination was later abandoned because it was suspected to cause a large increase in the computation time for the dynamic programming routine. This is because the changes in the revenue of the ballast trip are computed in the innermost loop of the dynamic programming algorithm. Each time the ballast trip revenue is computed, it is checked whether the current cargo pair is identical to the critical combination in the present node of the branch-and-bound tree or in any of the preceding nodes. The amount of work for this checking operation is thus proportional to the number of levels in the tree.

Next, the algorithm was run with a critical combination consisting of one vessel and one cargo, both represented in the critical matrix F_1. This combination is very easy to force in or out of the subprograms by changing the revenue of this critical cargo when the subprogram for the corresponding vessel is solved, and requires no extra computer time. It has been observed in one case, however, that the same cargo/cargo combination has caused similar fractional solutions even after several vessel/cargo branchings. Therefore, a third variant is presently tested, where a critical column with two cargoes in the critical matrix is selected. The branching is then performed on a vessel/cargo combination with one of the two critical cargoes and the vessel that carries the two cargoes in the critical column.

1.4 Computational Experience with Simulated Problems

In the previous paper it was reported that out of 900 simulated problems with 10 vessels and 15 cargoes, 14 problems had fractional solutions. Of 100 problems with 15 vessels and 25 cargoes, two solutions were fractional. All these 16 fractional problems were solved by the branch-and-bound

algorithm with one branching only. In one case the fractions in the original solution were ¼, and in the branch where the cargo combination was prohibited, the new solution was still fractional with fractions equal to ½. However, the other branch gave an integer solution with a higher value that terminated the tree. In the other 15 problems, the fractions were ½ and all the solutions along the branches were integer.

The 1000 simulated problems were run on a CD 3200 computer at the Royal Institute of Technology in Stockholm. Much larger problems than the ones with 15 vessels and 25 cargoes could not be run because of the limited core storage on that computer. When an IBM 360/75 was installed in Stockholm in 1968, the last series of 20 simulated problems with 40 vessels and 50 cargoes was run on that computer. The average solution time was 1.2 min. In all the simulated problems, the branch-and-bound algorithm used the cargo/cargo combination in the subprograms.

Five of those problems had fractional solutions, and four of those were solved after one branching. In the remaining problem, a more complicated tree occurred as shown in Fig. 2. In the first branching with the critical cargo combination 8/50, node 2 gives an integer solution with the value 867.9097. In node 3, the combination 7/22 is selected, and the created nodes 4 and 5 both have fractional solutions. Node 4 has the largest estimate, so a new critical combination 10/16 is selected in this node. The resulting nodes 6 and 7 are both fractional, but node 6 has a value that is smaller than the value of the best integer solution, which means that this node can be discarded. Node 7 has a slightly higher value than node 5, so a critical combination 32/45 is selected in node 7. The solutions in node 8 and node 9

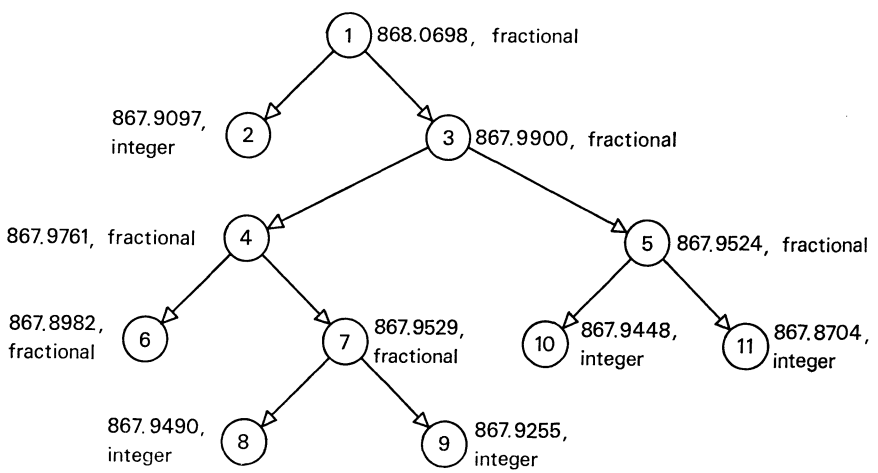

Fig. 2 Solution of the only simulated problem requiring more than one branching.

are both integer, and node 8 has the best value, 867.9490. Finally, a branching is made from node 5 on the combination 4/20. The two new nodes (10 and 11) are both integer and inferior to the solution in node 8, which thus is optimal. In this problem, more complex fractional solutions occurred which was obvious from the determinants that took the values 2, 3, 5, and 7 in different nodes. This showed for the first time that the method is capable of handling such complicated problems.

It is apparent from the simulation of problems with 40 vessels and 50 cargoes, where 25 per cent of the solutions were fractional, that the frequency of fractional solutions increases with problem size. As it is expensive to simulate a large number of full-size problems with about 100 vessels and 150 cargoes, the performance of the algorithm must be evaluated from the results obtained with the actual scheduling problems. These results are reported in section 3 below.

2. THE CUTTING PLANE ALGORITHM

2.1 Selection of a cut

During the simulation runs with small problems, it was observed that the fractional solutions in almost all cases were caused by a square staircase matrix with an odd number of rows like the one shown below.

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The determinant of such a matrix is ± 2; thus it causes a solution with the fractions $\frac{1}{2}$. Let x_j^s, $1 \leqq j \leqq J$, denote the variables corresponding to the columns in such a matrix. Then the constraint

$$\sum_{j=1}^{j=J} x_j^s \leqq \frac{J-1}{2}$$

is a cut that excludes the current solution where all $x_j^s = \frac{1}{2}$, but does not exclude any integer solution.

Thus, the cutting plane algorithm was based on the observation that most fractional solutions are caused by a staircase matrix. A search is made among the fractional-valued columns for such a matrix with an odd number of rows, after which the new constraint is added.

In order to generate a stronger constraint, it was considered desirable to prevent other columns in the LP problem to replace columns in the staircase matrix such that a similar fractional solution occurs again. Thus,

a list is made up over the row combinations in the staircase matrix. These combinations consist either of a vessel row plus one cargo row or of two cargo rows. The entire master LP problem is then searched for other columns having nonzero elements in any of the listed row combinations. Such columns are then included in the new constraint. It is easily seen that this extended constraint does not prevent any integer solution either.

$$\begin{bmatrix} 1^* & 1^* & 1 & 0 \\ 0 & 1^* & 1^* & 1 \\ 1^* & 0 & 1^* & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

The advantage of extending the constraint follows from the matrix above. This critical matrix causes the fractional solution $x_1 = x_2 = x_3 = x_4 = \frac{1}{3}$. A staircase matrix is found among the first three rows and columns, consisting of the row combinations 1-3, 1-2, and 2-3. This staircase matrix generates the constraint

$$x_1 + x_2 + x_3 \leqq 1,$$

but this is satisfied already in the present solution. As the fourth column contains the row combination 2-3, it is included in the extended constraint, which thus becomes

$$x_1 + x_2 + x_3 + x_4 \leqq 1.$$

The same constraint would be obtained by an ordinary cutting plane algorithm† and is not satisfied by the current solution.

In the case that a new fractional solution occurs in spite of the new constraint, more constraints can be generated by the same process. Because of the difficulties in implementing the new constraints in the subprograms, the previously generated constraints are not used in the search for a new staircase matrix. This may obviously lead to a situation where no such matrix can be found.

When a new constraint has been added to the master LP problem, the old solution is infeasible. A feasible solution is obtained by adding an artificial variable to the new constraint, after which this variable is minimized.

2.2 Implementation of a Cut in the Subprograms

In the preceding section, a cut was defined as a constraint with unit elements in all columns having certain pairs of nonzero elements according to a list of such critical pairs, which can consist either of one vessel row and one cargo row or of two cargo rows.

† *See*, for instance, Dantzig,[2] Chap. 26.

When a subprogram is solved in a decomposition algorithm, the objective function is modified by the dual variables in the master LP solution. In the cutting plane algorithm, we have new constraints in the master problem with unit elements for columns with certain vessel/cargo and cargo/cargo combinations. This means that the value of all solutions to subprograms where these combinations are present have to be corrected with the value of the dual variable for the new constraint.

Thus the dynamic programming algorithm has to be revised in order to take care of these value corrections. As was stated in section 1.3, it is easy to change the value for certain vessel/cargo or consecutive cargo/cargo combinations. The cutting plane algorithm was thus based on the assumption that the cargoes in a cargo/cargo combination are consecutive.

It is now clear why the new constraint has been constructed in the way it was done in the previous section. A staircase matrix is assumed because two-element combinations are the only ones that can easily be implemented in the subprograms. If more than one new constraint is generated, the previously generated constraints must not occur in the staircase matrix for the same reason. If the cargoes in a cargo/cargo combination are not consecutive, the problem cannot be handled with the presently used algorithm, but such a case can be treated if the subprograms are solved twice, as was stated in section 1.3.

2.3. Computational Experience

After it was established that the column generation algorithm gave fractional solutions in about 1–2 per cent of the cases for small problems, the above cutting plane algorithm was constructed and tested during early 1968. The tests were made on 12 fractional problems that originated from a series of 600 simulated problems with 10 vessels and 15 cargoes. All fractional values were equal to $\frac{1}{2}$. 10 of the 12 problems were solved with one constraint only. The other two problems could not be solved, because after a couple of constraints had been generated, a nonbinding constraint was generated. It was obvious then that more complicated matrices can be generated by the new constraints, making the staircase matrix method inefficient. At that stage, it was decided that the branch-and-bound approach should be tried, and as it turned out that all the test problems could be solved by that method, the cutting plane approach was abandoned.

3. EXPERIENCE WITH THE SCHEDULING MODEL

COMPARED TO THE computer program that was used when the previous paper was published, some important changes were made before the experiments with actual scheduling problems started during the summer of 1969.

First, the LP matrix was stored in packed form instead of the explicit

representation originally used in the LSUB routine. Thus, the matrix of a 700 × 200 problem requires only 5,600 words instead of the 140,000 that was previously required. We assume that no more than seven cargoes can be carried in one sequence, thus we reserve 8 words per column. The main requirement for core storage comes now from the inverse of the basis, which requires m^2 elements in a problem where the number of vessels and cargoes together is m.

Second, two different restart options were built in. The first one is used when the problem is not solved in the assigned computer time. The current basis and the entire LP matrix is then punched on cards, so that the algorithm can be started from the point where the computations were interrupted. The second restart option is used when only small changes in the data have occurred since the previous run. The columns of the optimal basis are punched for every optimal solution, and these cards are read together with the new data. It is then checked whether each old cargo sequence still is feasible, and if so, its new revenue is computed and the column is stored in the LP matrix.

When these changes had been made, the program was run with a realistic problem with 59 vessels and about 70 cargoes. The computation time on the IBM 360/75 was about 10–15 min for the column generating algorithm only and exceeded 30 min for a fractional problem. As this was quite disappointing, a test run was made on a CD 6600 at the recently opened CDC Center in Stockholm. As the running time was reduced by a factor of 2.5 and the total cost by a factor of 3, the program was converted to the CD computer. Soon after this, two other important program changes were made.

A. The number of columns that were retained in the master LP problem was increased to 900 from its old value, which was equal to the number of basic columns plus the number of vessels, i.e., about 200 for a 60 by 70 problem. By this simple change the running time was reduced by a further factor of 3, which must be due to the fact that the retained columns cause the dual variables to converge much faster.

B. If two or more cargoes can be treated as identical, the right-hand side for one cargo row can be set equal to the number of identical cargoes. This reduces the size of the LP problem and seems to have no negative effect on the frequency of fractional solutions. About 15 cargo rows are saved because of this change in a normal problem.

In September, 1969, a parallel test was carried out during one week when four slightly different problems were run. The problem size was reduced to approximately 65 vessels and 60–70 cargo rows by deletion of some traffic areas. The computation time ranged from 23 sec to 9 min, where the

maximum time occurred for a fractional problem with 11 nodes in the solution tree.

From December, 1969, full-scale problems are run with about 100 vessels and 120 cargo rows, i.e., about 135 cargoes. The planning period was cut down from 8 to 6–7 weeks in order to decrease the number of cargoes, since the core storage on the CD 6600 limits the problem size to 230 rows in the LP problem. The computation time varied during the eight December runs between 4 and 16 min for the decomposition algorithm, and was about 35 min for the only fractional problem that was solved. In total, 20 large-scale problems had been run by the end of 1969, nine of which had integer solutions. Of the eleven fractional problems, six were solved with the algorithm, four were infeasible, and one was not solved because of a program error. Infeasible runs are not pursued, because a schedule where some contracted cargoes are not shipped is of no interest. The number of nodes in the branch-and-bound solution varied between 5 and 11.

A value tolerance is used in the branch-and-bound algorithm such that an integer solution is accepted as optimal if the difference between its value and the value of the greatest upper bound in the tree is less than the tolerance. The tolerance that has been used during the runs reported above is about 60 per cent of the value of one day for an average vessel, which ought to be acceptable since the problem comprises more than 3000 vessel-days. If the value tolerance is increased, the number of nodes in the solution trees will be somewhat reduced.

From August 1970, the algorithm has been used regularly once or twice a week. It is operated directly by the officer in charge of scheduling via a Control Data Mark II terminal. Most of the runs are made with a four-week planning period, although eight- or even ten-week schedules are produced occasionally.

The computer-produced schedules are generally subject to manual revisions because of fine details concerning exact arrival hours, dockings, split loading or discharging, etc., which cannot be taken into account by the algorithm. The schedules are thus regarded more as a very good tentative plan than as a final product. It has not been possible to compare the efficiency between manual- and computer-produced schedules, but the latter are nevertheless highly appreciated because they eliminate a lot of the tedious task of the scheduling officer.

REFERENCES

1. L. Appelgren, "A Column Generation Algorithm for a Ship Scheduling Problem," *Trans. Sci.* **3**, 53–68 (1969).

2. G. B. DANTZIG, *Linear Programming and Extensions*, Princeton University Press, Princeton, N. J., 1963.
3. A. H. LAND AND A. G. DOIG, "An Automatic Method of Solving Discrete Programming Problems," *Econometrica* **28,** 497–520 (1960).
4. E. L. LAWLER AND D. E. WOOD, "Branch-and-Bound Methods: A Survey," *Opns. Res.* **14,** 699–719 (1966).
5. A. LEVIN, "Some Fleet Routing and Scheduling Problems for Air Transportation Systems," Flight Transportation Laboratory Report R-68-5, Massachusetts Institute of Technology, January 1969.

(Received, January 1970)