
Computational Optimization and Applications, 22, 351–367, 2002

An Integral Simplex Algorithm for Solving Combinatorial Optimization Problems

GERALD L. THOMPSON
Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract. In this paper a local integral simplex algorithm will be described which, starting with the initial tableau of a set partitioning problem, makes pivots using the *pivot on one rule* until no more such pivots are possible because a local optimum has been found. If the local optimum is also a global optimum the process stops. Otherwise, a global integral simplex algorithm creates and solves the problems in a search tree consisting of a polynomial number of subproblems, subproblems of subproblems, etc. The solution to at least one of these subproblems is guaranteed to be an optimal solution to the original problem. If that solution has a bounded objective then it is an optimal set partitioning solution of the original problem, but if it has an unbounded objective then the original problem has no feasible solution. It will be shown that the total number of pivots required for the global integral simplex method to solve a set partitioning problem having m rows, where m is an arbitrary but fixed positive integer, is bounded by a polynomial function of n.

A method for programming the algorithms in this paper to run on parallel computers is discussed briefly.

1. Introduction

In this paper a new approach to solving some kinds of zero/one integer programming problems will be presented which (a) is independent of previous popular approaches such as cutting plane or branch and cut methods; (b) depends only on linear algebra and linear programming; together with some new results contained in this paper; (c) requires only a polynomial number of pivots to solve any set partitioning problem having m rows, where m is an arbitrary but fixed positive integer; and (d) and provides a certificate of optimality for the final solution. The complete theoretical development of the method and preliminary computational results are presented.

The basic idea of the method of this paper is to use a modified version of the simplex method, called the *local integral simplex method*, in which pivot steps are made only on one entries in the simplex tableau. Since the problems considered here are set partitioning problems, the initial pivots on ones are easy to find. However, in later steps the integral simplex method frequently stops at a point at which there are columns having negative indicators which could improve the objective function if brought into the basis, but the minimum ratio rule, applied to these problems, can find pivots only on numbers larger than one. In other words the method finds only a local optimum for the problem. In order to find the overall optimum another method, called the *global integral simplex algorithm*, is applied which uses the local simplex method as a subroutine. Once the local method stops at a local optimum the global method defines a subproblem for each negative reduced "cost" in the local method's final tableau. It is shown

in this paper that the best solution found among all of the subproblems is the global optimum (if there is one) for the original problem. It is also shown that the number of pivot steps needed for the global integral simplex method to solve a set partitioning problem having m (where m is fixed) rows is bounded by a polynomial function of n.

In previous papers Balas and Padberg [1, 2] have shown that the simplex method, when restricted to pivoting using the pivot on one rule, can be used to explore the integer solutions within the feasible set of the linear programming relaxation of the set partitioning problem. In their 1975 paper they provide two algorithms for solving set partitioning problems with their pivoting method together with column generation techniques.

2. The local integral simplex method (LISM)

Many combinatorial optimization problems have one of the following forms:

$$\begin{aligned} &\text{Minimize} && cx \\ &\text{Subject to} && Ax \leq, =, \geq b \\ & && x \in \{0,1\}^n \end{aligned}$$

where A is an $m \times n$ matrix with 0/1 entries, x is an $n \times 1$ column vector, b is an $m \times 1$ vector with positive integral entries, and c is a $1 \times n$ vector with positive integral entries.

For instance, the *set partitioning problem* can be stated as:

$$\begin{aligned} &\text{Minimize} && cx + My = z \\ &\text{Subject to} && Ax + y = f \\ & && x \in \{0,1\}^n \\ & && y \geq 0 \end{aligned} \tag{1}$$

where A is an $m \times n$ 0/1 matrix, x is an $n \times 1$ vector, y is an $m \times 1$ vector whose components are called *artificial variables*, c is an $1 \times n$ vector of integer costs, f is an $m \times 1$ vector of all ones, M is a large positive number and scalar z is the value of the objective function. All the example problems considered in this paper will be set partitioning problems. Solutions (x, y) to this problem have coefficients 0 or 1, and are called *partition solutions*. Since the integral simplex method uses the pivot on one rule all of the entries in the tableaus calculated by this method are integers. A partition solution (x, y) is said to be an *integral partition* if $y = 0$.

In the LP relaxation of the problem, the condition "$x \in \{0,1\}^n$" in (1) is replaced by the condition "$x \geq 0$". The integral partition solutions just defined, found by the usual simplex method, are extreme points of X, the set of all feasible solutions to the LP relaxation. Moreover, X has other extreme points, called *fractional partitions*, (x, y), whose components are real numbers in the closed interval [0,1], and having at least one component x_j satisfying $0 < x_j < 1$.

x_1	x_2	x_3	x_4		
1	0	1	1	1	y_1
0	1	0	1	1	y_2
0	1	1	0	1	y_3
-9	-19	-19	-19	-30	

(a)

y_1	y_2	y_3	x_4		
1	1	-1	2	1	x_1
0	1	0	1	1	x_2
0	-1	1	-1	0	x_3
9	9	10	-1	-2	

(b)

Figure 1.

We use the following notation for the tableaus of the problem. Let T be the $(m + 1) \times (n + m + 1)$ initial tableau of a set partitioning problem in which the $m \times n$ matrix A is put in rows $1, \ldots, m$ and columns $1, \ldots, n$, the artificial vectors are put in rows $1, \ldots, m$ and columns $n + 1, \ldots, n + m$, the b vector is put in column 0 in rows $1, \ldots, m$, the c vector is put in row 0 in columns $1, \ldots, n$, and the costs M of the artificial variables are put in row zero and columns $n + 1, \ldots, n + m$. In row 0 and column 0 a zero entry is placed which will record the values of the objective function. In the LP phase I step the artificial vectors are pivoted in, to become the initial basis, and the last m columns are dropped to give a compact initial tableau, see figure 1(a). The entries in row 0, columns $1, \ldots, n$ are called *indicators* of the corresponding variables that label the columns, instead of reduced costs as in ordinary LP, because there are *no dual variable prices*. (However, no harm is done by thinking of them as reduced costs). The indicators of the basic variables whose labels appear on the righthand side of the tableau are all *zeros*.

The steps of the local integral simplex method, written in a pseudo C language, are shown in figure 2. In that figure there are two places where a RULE is referred to. By that is meant one of the following rules: Bland's rule for preventing cycling, the lexicographic ordering rule for preventing cycling, or random selection rule for preventing cycling with probability one, see, for instance Chvatal [3].

When LISM stops its final tableau may have all nonnegative indicators (i.e., reduced costs) which means that a global optimum has been found. If LISM has some negative indicators, then it is possible that there are some better integral solutions. The global integral simplex method to be illustrated next, and covered fully in Section 3, will find such better solutions. To do so it defines a subproblem at a node of the tree having a negative indicator x_j by calculating the effect in the tableau of making x_j equal to one and omitting that variable in the tableau of the new subproblem. In addition none of the variables at the node of the tree that were previously set to one are included in the new subproblem (see figure 3). It then sets up and solves each of these subproblems (which in turn may require solving subproblems of subproblems, etc.) so defined.

To illustrate, consider the 3×4 example whose A matrix and c vector are shown in figure 4.

Putting these quantities and some artificial variables into the initial (infeasible) 3×7 simplex tableau shown in figure 5. In that figure the primal variables are the x's and the artificial variables are the y's. Note that the "large" cost for the artificial variables is $M = 10$. We follow the usual convention that the entries of the c vector and the penalty costs M in (1) are negatives, and the entries in the last row of the first 7 columns are their negatives.

```
SET UP:
    Create the initial tableau of the problem and perform phase I as described
    Let T be the resulting tableau with basic and non basic variables labeled
PIVOT STEP:
CHOOSE PIVOT{
    Make a list L of columns of T having negative indicators
    If (L is not empty){
        Apply RULE to select and remove an element j of L
        Find minratio = minimum t_i0/t_ij
                        t_ij > 0
        Make a list P of rows i with pivot element t_ij=1 and t_i0=minratio
        If (P is not empty){
            Use RULE to select and remove an element of P
            Go to UPDATE
        }
    }
    goto HALT
}
UPDATE{
    Solve the problem using the usual updating steps of the simplex method
    Go to PIVOT STEP
}
HALT: /* A locally optimal solution has been found */
```

Figure 2. Pseudo C code for LISM.

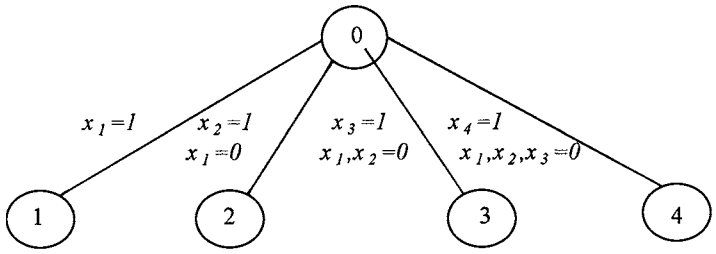

Figure 3.

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad c = (1 \quad 1 \quad 1 \quad 1)$$

Figure 4.

$A =$

x_1	x_2	x_3	x_4	y_1	y_2	y_3	
1	0	1	1	1	0	0	1
0	1	0	1	0	1	0	1
0	1	1	0	0	0	1	1
1	1	1	1	10	10	10	0

Figure 5.

Pivoting in each of the artificial variables gives the initial compact simplex tableau shown in figure 1(a). Pivoting on the entries (1,1), (2,2), and (3,3) in figure 1(a) gives the simplex tableau of the local optimum shown in figure 1(b). No further pivoting is possible even though the fourth column has a negative indicator of -1, because when the minimum ratio rule is applied to column 4, it chooses the 2 entry in the (1,4) location on which to pivot. At this local optimum the current integral solution is given by $x_1 = x_2 = 1$ with objective function $z = 2$. The sign of the objective is changed because the problem requires minimizing the sum of the negative costs. At this point the LISM algorithm stops because no further pivots on one can be made.

In Section 3 the solution given in figure 1(b), $x_1 = x_2 = 1$ and $z = 2$, is shown to be optimal for the problem in figure 5.

It is easy to show from figure 5 that there is another *fractional* solution to the problem, namely $x_2 = x_3 = x_4 = 1/2$ with $z = 3/2$. This can also be seen by making an ordinary LP pivot on the 2 entry in row 1 and column 4 of figure 1(b). This solution is, in fact, a *fractional* solution obtained as the linear programming solution to the LP relaxation of the integer problem given in figure 5.

3. The global integral simplex method (GLISM)

We begin this section by making some definitions.

Definition 1. Two zero/one column vectors a and b with the same number of components are said to be *orthogonal* if their inner product $a \cdot b$ is zero, i.e., if for every row i the product $a_i b_i = 0$. They are said to be *non-orthogonal* if for some row i the product $a_i b_i = 1$.

For instance, in figure 1(a) columns 1 and 2 are orthogonal while column 4 is not orthogonal to any of the columns 1, 2 or 3. So that if we wish to set x_4 to 1 we would also have to set $x_1 = x_2 = x_3 = 0$ to create a feasible solution.

Definition 2. Let T be the tableau of the original problem, let T_j be the jth column of T and let N be the indices of an orthogonal set of columns of T that are to be set equal to 1; let Z be a set of columns that are to be set equal to 0; then the subproblem defined by setting $x_j(1) = 1, \ldots, x_j(k) = 1$ is obtained by doing the following steps: (a) let $w = \sum_{j \in N} T_j$; and cross out non-orthogonal columns j of T for which $w \cdot T_j > 0$; (b) cross out columns j of T which belong to Z, (c) calculate the effect of setting the columns in N to one by pivoting on a one for each element of N; (d) cross out rows i of T such that $w_i = 1$; and

(e) cross out all rows and/or columns of T that have all zero entries. The resulting tableau, if not empty, is called a *subproblem* of the original problem.

When crossing out a column, the label at the top of the column must also be eliminated. Thus, when crossing out columns 1, 2, and 3 in figure 1(a), the labels x_1, x_2, and x_3 are also crossed out. Note that in setting a variable to zero, no record of that variable needs to be kept since the absence of that variable in any solution found subsequently indicates that it is zero.

In figure 1(b) it is clear that to get a better solution the variable x_4 must be set to 1 and eliminated from the tableau. But this means that columns x_1, x_2, and x_3 must also be crossed out which means that the whole tableau is eliminated and no subproblem has been created. This proves that there is no better integral solution than the one shown in figure 1(b), which is therefore optimal.

Definition 3. Let T be the final tableau of a set partitioning problem or subproblem, and let x be the partition solution shown in T for the problem. If x^*, not equal to x, is feasible partition solution, $x_j^* > 0$ is a nonbasic variable labeling a column of T having a negative indicator, then x_j^* is said to be an *set partition flag variable* for x^*. If x^* is an integral partition, then x_j^* is said to be an *integral set partition flag variable for* x^*. If x^* is a fractional solution then x_j^* is said to be a *fractional set partition flag variable for* x^*.

As an example, note that in final tableau of figure 1(b) the variable x_4 is a fractional set partition flag variable for the fractional solution $x_2 = x_3 = x_4 = 1/2$ and $z = 3/2$.

It should be remarked that a set partition flag variable appearing in a final tableau can simultaneously be an integral flag variable for one or more integral set partition solutions and also a fractional set partition variable for one or more fractional set partition solutions. This will be illustrated in the numerical example of figure 6 in Section 3.

The global integral simplex method GLISM constructs a search tree of subproblems which are solved using LISM as a subroutine. To see how the search tree is constructed see figure 3. In that figure node 0 stands for any new node (including the initial node in the tree) that is to be added to the tree, nodes 1, 2, 3, and 4 stand for subsequent nodes, and the symbols x_1, x_2, x_3, x_4 (etc.) stand for the values of the new variables having negative indicators in the current tableau.

1	2	3	4	5	6	7	8	9	10	11	
0	1	0	0	0	1	1	0	1	1	0	1
1	0	0	0	1	1	0	0	1	1	1	1
1	0	1	0	1	1	0	1	0	1	0	1
1	0	1	1	1	1	0	0	0	0	0	1
0	0	1	1	1	0	1	0	0	0	0	1
72	48	77	44	56	49	77	41	47	96	42	0

Figure 6. Data for a 5×11 example.

```
BEGIN:
        Make a copy T of the tableau of the original problem
START{/* Create the tableau of problem P_k = (k, f_k, N_k , Z_k ) */
        Set the vector w = T_j^(1) + T_j^(2)+ ...+ T_j^(k)
        Cross out all non-orthogonal columns j of T for which the inner product w · T_j > 0
        Cross out all of the columns whose indices are in the Z_k list.
        Cross out all rows h of T that have w_h=1
        Eliminate any zero columns in T
}
SOLVE{
        Using LISM, solve the problem in T, saving its solution if it is the best so far
        Add to the list Q all of the new subproblems created by the negative indicators in
        the final tableau T^f
}
SELECT{
        If (Q is empty) goto HALT
        Else choose and remove a problem P from Q
        Go to START }
HALT: /* The best solution found is optimal */
```

Figure 7. Pseudo C code for GLISM.

To create the tableau at node 1, the effect of setting x_1 is calculated and the column labeled x_1 is crossed out. To calculate the tableau at node 2, the effect of setting $x_2 = 1$ is calculated and the columns labeled x_1 and x_2 are crossed out. To create the tableau at node 3, the effect of setting $x_3 = 1$ is calculated and the columns labeled x_1, x_2, and x_3 are crossed out. Similarly for node x_4. Note that due to the crossing out of columns, the sizes of subproblems tend to decrease rapidly as the depth of search increases, making them easier to solve.

It will be shown that the optimal solution to the subproblem having the smallest objective function value is also an optimal solution to the original problem, provided that problem has an optimal solution.

A pseudo C code for the global simplex method is given in figure 7. In that figure N is the subset of variables that must be set equal to one; Z is the subset of variables that must be set equal to zero, and Q is the subset of problems that are still to be solved.

4. Theoretical results

Here we will derive all of the theoretical results that are needed to prove that the GLISM algorithm will find optimal solutions (if such exist) to set partitioning problems. Although some of the results are similar to those that are true for the theory of linear programming, others are not.

Property 1. *Let T be the tableau of a locally optimal integer partition solution x.*
(a) *The objective value of x is the negative of the entry in the first row and first column of T.*
(b) *If x^* is any other locally optimal integral partition then its objective value is the objective value of x shown in T plus the sum of the indicators of T corresponding to the components of x^* that are equal to* 1.

Property 1(a) follows because the problem being solved by the simplex method is to maximize the negative of the objective function. Property 1(b) follows directly from the *invariance of linear relations property* which is: *any linear relation that holds for vectors in one basic solution holds for the same vectors in every other basic solution*. The latter holds because linear transformations, such as the pivot transformations of the simplex method, preserve linear relations.

Theorem 1. *Let T be the tableau of a locally optimal integer partition x which has some negative indicators and assume there is a better locally optimal integer partition x^*. Then*
(a) *T contains a column labeled with a nonbasic variable x_j having a negative indicator, and such that $x_j^* = 1$; i.e., x_j is a set partition flag variable for x^*.*
(b) *x^* is a locally optimal integer partition for the subproblem of the current problem which is defined by setting x_j equal to one.*

Proof:

(a) Let x and x^* be partition solutions having the properties stated in the hypothesis of the theorem. By Property 1(b) the objective function of x^* is equal to the sum of the objective function of x plus the indicators of components of x^* for which $x_j^* = 1$. Since x^* has a smaller objective value that x does, at least one of the indicators of x^* for which $x_j^* = 1$ must be negative. Since the indicators of the basic variables are always equal to 0, it follows that x_j^* must be nonbasic.
(b) From the assumptions, $x_j^* = 1$ in x^*. Also the column labeled x_j is orthogonal with the column of every other variable x_k^* that equals 1 in x^*, so that the subproblem defined by $x_j = 1$ will contain all the variables that are equal to one in x^*, except for x_j itself. Since x^* is optimal in the original problem it will also be optimal in the new subproblem. For suppose on the contrary, that x^{**} is better than x^* in the subproblem. Then all column vectors h, with $x_h^{**} = 1$ in the subproblem must be orthogonal with x_j^*, which means that x^{**} must have been better than x^* in the original problem, a contradiction. □

Corollary 1. *Either the current solution is globally optimal for the current problem, or else T will have a column whose label is a set partition flag variable for such a globally optimal solution x^0.*

The result in the Corollary 1 is much stronger than any similar result for linear programming. In the simplex solution of an LP problem it rarely happens that a variable having a negative indicator at some point during solution process will belong to a globally optimal

solution. But in the case of LISM it happens for at least one variable in every tableau whose solution is locally but not globally optimal.

Theorem 2. *Let T be the tableau of a locally optimal partition x which has some negative indicators and assume there is a better locally optimal fractional partition x^*. Then*

(a) *Tableau T contains a column whose label x_j^* is a fractional set partition flag variable for x^*.*

(b) *If x_j^* is an integral variable of x^*, i.e., $x_j^* = 1$, then x^* is a locally optimal fractional partition for the subproblem that is defined by setting $x_j^* = 1$.*

(c) *Let $x_j^* < 1$ be a fractional variable of x^*; then x^* is not a locally optimal partition for the subproblem that is defined by setting $x_j^* = 1$.*

The proofs of (a) and (b) are similar to the proofs of the previous theorem.

To prove part (c), recall that if x_j^* labels column j and is a fractional variable then there is at least one other column which is labeled by a fractional variable x_k^* and, since $x_j^* < 1$, for these two columns there is at least one row in which they both have a 1, i.e., they are non-orthogonal. Hence the subproblem P defined by setting x_j^* equal to 1 does not contain the column labeled by x_k^* which means that the fractional solution x^* is not feasible for problem P.

Property (c) of this theorem is the key result that shows how the global simplex method eliminates fractional partitions as candidates for globally optimal integer partitions by defining subproblems as the search tree is being created. Notice that when the local integral simplex method stops at a local minimum the global integral simplex method merely searches for negative indicators in the tableau and sets the corresponding variable equal to one to create a new branch in the subproblem search tree without determining whether that variable corresponds to (a) an integral set partition, or only to (b) a fractional set partition. In the case (a) the tree may branch further to locate other, and perhaps better integral solutions, while in case (b) the fractional partition solution simply vanishes, and does not appear anywhere lower in the subproblem search tree.

Theorem 3. *Assume that the final tableau T of a problem (or subproblem) has all non-negative indicators.*

(i) *If all artificial variables have value zero in the final tableau, the solution displayed in that tableau is an optimal set partition for the problem.*

(ii) *If one or more artificial variables have value one in the final tableau, the problem has no feasible set partition.*

Proof:

(i) Under the stated assumption, by Property 1(b) every other feasible partition has a cost greater than or equal to that of the solution shown in T. Since the displayed solution is feasible it therefore is globally optimal.

(ii) Under the stated assumptions it follows from Property 1(b) that the objective function is greater or equal to M, and by Property 1(a), all other solutions have objective value greater than or equal to M. Since M can be arbitrarily large, and since feasible solutions have finite values, it follows that there is no feasible partition. □

Theorem 4. *For any set partitioning problem the global integral simplex method will either find a globally optimal integral set partition, or else prove that no such feasible partition exists.*

Proof: The convergence of LISM when used to solve subproblems can be proved as in the simplex method by using the well known properties of the RULE in figure 2. By Theorem 2, either the set partition which appears in the final tableau of the first subproblem solved, or one of the set partitions whose flag variable appears in the tableau at a local optimum of one of the subproblems, is a globally optimal solution. Since the global integral simplex method finds an optimal solution for every subproblem corresponding to negative indicators in the final tableau, the subproblem yielding the best locally optimal solution is therefore globally optimal. If that solution does not contain any artificial variable with value one then it is a globally optimal set partition. If that solution does contain one or more artificial variables with value one, then by Theorem 3(b) the original problem does not have any feasible partition. □

The next theorem is not needed for later developments in this paper. However it is included because it settles an important question.

Theorem 5. *A set partitioning problem has an optimal integral dual solution if and only if the linear programming relaxation of the original problem has an optimal integral primal solution.*

Proof: Assume that the linear programming relaxation of the original problem has an optimal integral primal solution. Then applying GLISM to that problem such an optimal solution whose dual is also optimal and integral will be found. □

If the linear programming relaxation does not have an optimal integral primal solution then there is a gap between the integral optimal solution and the linear programming solution. Moreover the latter solution must be fractional by the assumption of the theorem. Let X be the optimal integral primal solution found by GLISM. Because the LP relaxation is a fractional solution having a lower cost objective function, the final tableau for X will have at least one negative indicator, indicating that there is no feasible dual solution for X.

5. Computational complexity of the algorithms

In this section it is shown that both LISM and GLISM can be solved in a finite number of pivots when used to solve set partitioning problems having a fixed number m of rows, by showing that the maximum number of pivots required to solve either of them is bounded by a polynomial function of n.

Theorem 6. *The maximum number of pivots needed by LISM to find a locally optimal solution to a set partitioning problem having m rows and n columns is at most* $(n+1)^m$.

Proof: At each pivot LISM moves from one feasible basis having m columns to another. Let S be the set, having $m + n$ elements, that consists of the m artificial column vectors together with the n column vectors of A. Then the maximum number of pivots required for LISM to converge to a local optimum is less than or equal to the maximum number of m element subsets of S, which is the combinatorial coefficient

$$\begin{aligned}\binom{n+m}{m} &= (n+m)!/(m!\,n!) = (n+1)\cdot(n+2)\ldots(n+m)/m! \\ &< (n+1)\cdot((n/2)+1)\ldots((n/m)+1) \\ &< (n+1)^m\end{aligned}$$

which gives the upper bound asserted in the theorem. □

Theorem 7. *If $n \geq 2$ the total number of pivots required by the GLISM to solve an $m \times n$ set partitioning problem is less than $2(n^2+n)^m$.*

Proof: The original problem at level 0 of the subproblem search tree has at most n negative indicators, and hence it creates at most n subproblems at level 1 of the search tree. Each of these subproblems is of size at most $(m-1) \times (n-1)$, and hence creates at most $n(n-1) < n^2$ subproblems. Similarly, the size of subproblems on the kth level of the tree is at most $(m-k) \times (n-k)$ and hence they create at most $n(n-1)\ldots(n-k) < n^k$ subproblems. The deepest level of the tree is $(m-1)$, since "subproblems" on the level m would have no rows. Hence the number of subproblems on the deepest possible level is at most $n(n-1)\ldots(n-m+1) < n^m$. Adding these together, and using the fact that $n \geq 2$ implies $n/(n-1) \leq 2$, gives the total number of subproblems that must be solved as

$$n + n^2 + \cdots + n^m = n(n^m - 1)/(n-1) < 2n^m.$$

Multiplying this result by the result of Theorem 6 gives the bound stated in the theorem. □

6. Numerical examples

Consider the 3×4 problem shown in figures 4 and 5. In the final tableau of the problem shown in figure 1(b) note that the variable x_4 is the only flag variable. Observe that the fourth column in the A matrix is non-orthogonal with each of the other columns of the A matrix. Hence the subproblem created by setting $x_4 = 0$ consists of only the fourth column plus the artificial variables. Since the fourth column in A has a zero row entry, this subproblem has no finite solution. Therefore the solution shown in figure 1(b) is the optimal solution for the problem in figure 5.

Figure 6 contains the data for a 5×11 numerical example. In order to solve the problem using LISM, five artificial vectors are added, and a phase I start is made to get an initial feasible basis. After pivoting seven times and eliminating the artificial vectors the final compact tableau shown in figure 8 is obtained. The set partitioning solution shown in figure 8 is $x_1 = x_7 = 1$ with objective value $z = 149$.

x_5	x_3	x_4	x_9	x_{10}	x_2		
1	1	1	0	0	0	1	x_7
0	-1	-1	1	1	0	0	x_{11}
0	0	-1	0	1	0	0	x_8
-1	-1	-1	1	1	1	0	x_6
2	2	2	-1	-1	-1	1	x_1
-116	-53	-45	28	36	71	-149	

Figure 8. Local optimum for the 5 × 11 example found by LISM.

We see from the figure that there are three flag variables, x_3, x_4, and x_5 each of which defines a subproblem, as follows:

(i) Setting $x_3 = 1$ in the original problem data in figure 6 gives the 2 × 3 subproblem with rows 1 and 2, and columns 2, 9, and 11. The optimal integral solution to this subproblem (together with $x_3 = 1$) is: $x_3 = x_9 = 1$ and $z = 124$.
(ii) Setting $x_4 = 1$ gives the 3 × 5 problem with rows 1, 2, and 3, and columns 2, 8, 9, 10, and 11. The optimal integral solution to this subproblem (together with $x_4 = 1$) is: $x_4 = x_8 = x_9 = 1$ and $z = 132$. There is another nonoptimal solution to this problem, namely, $x_4 = x_{10} = 1$ and $z = 140$ which was not found by LISM since it found the best one first. Both of these solutions had x_4 as an integral set partition flag variable.
(iii) Setting $x_5 = 1$ gives a 1 × 1 problem with row 1 and column 2. The optimal integral solution to this subproblem (together with $x_5 = 1$) is: $x_2 = x_5 = 1$ and $z = 104$.

Since all three subproblems yielded optimal integral solutions there were no more subproblems to be solved, and the best of the solutions, given in (iii), is optimal for the problem in figure 8.

7. Certificate of optimality

In ordinary linear programming theory the optimal dual solution gives a *certificate of optimality* in the form of a bounding hyperplane P, the direction of whose normal is given by the optimal dual solution. This hyperplane contains at least one point of the convex set of feasible primal solutions X, namely, the optimal solution found by solving the linear programming problem, and all of the other feasible solutions in X lie either on the same side of P or actually on the hyperplane P.

It is possible to find something similar here, but not by using a dual solution to the problem, because, as indicated in Theorem 5, there is usually no feasible dual problem in the case of set partitioning. Suppose that we somehow knew of the feasible solution for figure 6, $x_2 = x_5 = 1$ and $z = 104$, and wanted to see if it were optimal. We start with the original problem in figure 8, add the artificial vectors and perform the Phase I step; then we pivot to bring into the basis the columns of the known solution, namely, the columns labeled x_2 and x_5. After that is necessary to pivot several more times to arrive at the locally optimal solution shown in the full tableau of figure 9.

x_1	x_3	x_4	x_{11}	x_{10}	x_6		
-1	0	0	0	0	-1	0	x_7
0	-1	-1	1	1	0	0	x_9
0	0	-1	0	1	0	0	x_8
1	1	1	-1	0	2	1	x_2
1	1	1	0	0	1	1	x_5
45	20	28	43	8	-26	-143	

Figure 9. Final tableau after pivoting in the columns labeled x_2 and x_5 and reoptimizing.

Note that in that figure the optimum solution is as found in (iii) above; however there is a negative indicator with label x_6. As usual we set $x_6 = 1$ and find that the corresponding subproblem has no columns since x_6 is not orthogonal with any other column in figure 6. Hence this subproblem does not have any finite integral solution which provides a second proof that the solution $x_2 = x_5 = 1$ and $z = 104$ is optimal. (The first proof was given by finding the solutions to the three subproblems in (i), (ii) and (iii) above.)

This is the certificate of optimality for the problem at hand. In the general case the procedure for finding the certificate of optimality is the same once an optimal solution is found by the GLISM. In order to find a certificate of optimality, set up the original tableau, and pivot on ones until the variables in the optimal solution become basic, then continue pivoting until a local optimal solution for the tableau is obtained. Then show that, for each negative indicator in the final tableau, the corresponding subproblems do not have better integral solutions. Note that here the certificate of optimality is not obtained as an automatic by product of finding an optimal solution to a linear program. It requires additional optional computations which should be done only if it is felt that the results might be useful, for instance, in interpreting the optimal solution.

In order to find the solution that corresponds to the flag variable x_6 in figure 9 we temporarily drop the pivot on one rule in order to return to ordinary linear programming, and pivot on the 2 entry in row 4 and column 6 in figure 9. We obtain the optimal *fractional* solution $x_5 = x_6 = x_7 = 1/2$ and $z = 91$. This is the LP relaxation solution of the problem shown in figure 6. What this means is that the flag variable x_5 back in figure 6 was both an integral flag variable for the optimal solution found in (iii), and also a fractional flag variable for the LP relaxation solution just found. When x_5 became basic with value one in figure 9 it could no longer be a flag variable for the fractional solution of the LP relaxation, so that role was taken over by x_6.

8. Computational experience

Preliminary computations were performed with a PC running at 33 megaherz. Table 1 contains numerical data from the solutions of 25 problems. Each row in the table is the average of solution times, densities and other data for 5 instances of randomly generated problems having 15 rows and 150 columns. Notice that the solution times decrease rapidly as the density of the problems decrease from 0.20 to 0.40. Similarly the numbers of subproblems and pivots also decrease rapidly. However the maximum length of the current subproblem queue

Table 1. Data from the solutions of 25 problems of size 15 × 150 having varying densities.

Density	Objective	Seconds	Sub-probs.	Pivots	Max Q	Depth
0.20	3.60	203.20	11,010	94,389	118.40	8.00
0.25	6.75	83.80	4,807	35,892	126.25	7.25
0.30	9.80	16.60	747	4,816	111.60	6.00
0.35	9.60	7.00	259	1,547	109.00	4.80
0.40	26.80	3.40	91	469	109.00	3.40

Table 2. Average solution times for a series of 160 randomly generated problems each having density of 0.3.

n	$m = 10$	$m = 15$	$m = 20$	$m = 25$
50	0	0	0	0
100	1.2	3.4	2.2	0
150	5.4	20.2	13.4	5.4
200	8.2	49.8	47.2	21.8
250	19.8	114.6	126.6	49.0
300	19.6	116.8	220.0	111.2
350	38.4	148.1	453.6	194.4
400	42.6	161.8	641.4	305.4

(Max Q) stays almost constant. The maximum depth of the subproblem search tree decreases slowly with the increase of problem density. Forty percent dense problems are easy to solve and have large objective functions because each column has only a few columns that are consistent with it and hence these problems have only a small number of feasible solutions.

Table 2 contains a listing of the run times (on a 33 megaherz PC) for problems having densities of 0.3 and sizes $m = 10, 15, 20, 25$ and $n = 50, 100, 150, 200, 250, 300, 350, 400$. Each entry is the average of 5 run times for each problem size instance. Note that the 20 row problems usually take the longest. It is not clear why 20 row problems should take longer than 25 row problems.

In the fall of 1999 T. Smith and the author wrote a new and much improved version of the GLISM code and used it to solve the practical set covering problems used by Hoffman and Padberg [4] to test their branch and cut algorithm. These problems were set partitioning problems having 20 to 35 percent densities arising from actual crew scheduling problems supplied by several different airlines companies. K. Anderson simplified the data by eliminating duplicate columns from the data supplied by Hoffman and Padberg. Some of the problems were relatively small and others much larger. In Table 4 the problem size data for 25 of these problems are shown together with solution times (HP) by the Hoffman and Padberg code and by our 1999 (ST) code. The computer we used for the (ST) times in Table 3 was a PC running at 333 megaherz, which is 10 times faster than the PC times reported in Tables 1 and 2. The computers used for the (HP) times in Table 3 were of 1992–93 vintage,

Table 4. Times to solve all of the smaller Hoffman-Padberg (HP) and some of the larger crew scheduling problems reported in [4] using their 1993 code and the Smith-Thompson (ST) 1999 code.

Problem identity	*m*	*n*	HP (sec)	ST (sec)
199	17	177	0.06	0.00
294	18	251	0.17	0.01
404	19	336	0.21	0.01
467	29	405	0.10	0.02
577	25	421	0.30	0.01
619	23	521	0.34	0.03
677	25	565	0.19	0.03
685	22	536	0.62	0.02
711	18	423	0.34	0.05
770	19	639	0.19	0.01
771	21	468	0.34	0.02
899	20	718	0.30	0.02
1072	17	982	0.38	0.06
1079	23	795	0.99	0.04
1210	18	582	0.40	0.01
1217	20	844	0.62	0.02
1220	21	723	1.35	0.03
1355	22	817	0.28	0.03
1366	19	926	0.56	0.02
1709	23	1191	0.48	0.05
1783	20	1408	3.68	0.06
2540	18	2034	0.99	0.11
2653	26	1877	0.75	0.08
2662	26	1823	1.43	0.15
3068	23	2415	1.45	0.07
6774	50	5977	10.41	0.83
8820	39	6488	2.08	0.33
16043	51	10950	4.29	1.60
28016	163	6564	11.19	141.65
nw04	35	46189	2,642.00	289.07

and were probably in the 100–200 megaherz range. The run time comparisons between the two different methods is thus very difficult. The faster time for ISM on the nw04 problem is probably due to the weakness of the linear cut used by HP. However, it appears that overall these two methods exhibit competitive performances in solving these problems.

T. Smith and the author are currently preparing a paper that gives a much more detailed and comprehensive comparison of the two methods on the data in [4].

The author and T. Smith have also written a version of the GLISM code suitable for running on parallel computers. At the initial node of the search tree this code assigns different parts of that search tree to different processors to solve. During the tree search, each processor broadcasts to each of the other processors, any improvement it finds on the upper bound to the solution objective function. This knowledge frequently reduces the overall search time. The parallel code has exhibited superlinear speedups for certain kinds of such problems; that is, doubling the number of processors used to run the parallel version of GLISM sometimes more than doubles its solution speed. This will be reported on in the forthcoming paper referred to above.

We also solved a number of randomly generated problems of the same sizes as those in Table 4. Surprisingly the run times for those random problems was considerably *longer* than for the practical problems in Table 4! This contradicts a commonly held belief that random problems are always much *easier* to solve than real life problems.

It is well known that empirically the computational difficulty of solving a linear program having a fixed number of rows m, is linear in m, e.g. approximately $3m$ or $4m$, and that is what makes the ordinary simplex method be so successful in solving practical problems. The computations indicate that, empirically, the time required to solve set partitioning problems with GLISM is a low degree function of n, so that this code, especially in its parallel version, can be expected to be quite successful in solving real applied problems. Of course this assertion must be confirmed by other researchers before a final judgment can be made.

9. Conclusions

In this paper two simple algorithms, which are based on adding the pivot on one rule to the simplex method together with the creation of a search tree of subproblems, have resulted in a zero/one integer programming method that, for fixed m, requires only a polynomial number of pivots either, to find an optimal solution to a given set partitioning problem, or else to prove that no feasible partition exists for the problem. It was also shown empirically that the method is likely to be successful in practice and offers considerable hope of being a useful tool for solving a large variety of practical combinatorial optimization problems.

The author is currently testing the same method for solving other combinatorial problems such as set covering, set packing, matching, independent set, maximum clique, postman and other problems.

Acknowledgments

The author would like to thank Egon Balas, Robert Carr, and Kent Anderson for their helpful comments during the preparation of this paper.

The work of the author was part of the activities of the Management Sciences Research Group, Carnegie Mellon University, under contract No. N00014-85-0198 NR 047-048 with the Office of Naval Research. Reproduction in whole or in part is permitted for any purpose of the US Government.

References

1. E. Balas and M. Padberg, "On the set covering problem," Operations Research, vol. 20, pp. 1153–1161, 1972.
2. E. Balas and M. Padberg, "On the set covering problem II: An algorithm for set partitioning," Operations Research, vol. 23, pp. 74–90, 1975.
3. V. Chvatal, Linear Programming, Freeman: San Francisco, CA, 1983.
4. K.L. Hoffman and M. Padberg, "Solving airline crew scheduling problems by branch-and-cut," Management Science, vol. 39, pp. 657–682, 1993.