The above definitions are illustrated in the following example.

COLUMN-COUNT = 2 2 2 1
f = 50 30 15 5

	R_1	R_2	R_3	R_4	WDC	DELTA	DASH-COUNT
C_1	-	Y	-	N	130	1	4
C_2	Y	N	Y	N	0	1	0
C_3	N	-	Y	Y	60	1	2

STEPS IN THE ALGORITHM

Step 1. Same as for Algorithm 1.

Step 2. Determine those rows that have a minimum weighted dash-count.

Step 3. If two or more rows have a minimum WDC, select from among them the row that has the minimum delta. If among these there still exists two or more rows, select the row with the minimum dash-count. If there are more than two such rows, select any one of them. The test on dash-count does not affect computer running time, but can save memory space without adding to computer running time.

Since all rows in the example have the minimum weighted dash-count, test their deltas. Since C_2, C_3 and C_4 have the minimum delta, test their dash-counts. Since all three eligible rows (2, 3 and 4) have the minimum dash-count, select any one of them as C_k.

COLUMN-COUNT = 4 4 4 4
f = 40 20 20 20

	R_1	R_2	R_3	R_4	WDC	DELTA	DASH-COUNT
C_1	-	Y	N	Y	160	4	
C_2	Y	-	N	-	160	0	8
C_3	Y	N	-	-	160	0	8
C_4	Y	-	-	N	160	0	8
C_5	-	Y	Y	N	160	4	

Steps 4, 5 and 6. These steps are the same as for Algorithm 1.

The procedure for Algorithm 2 is illustrated in Figure 2.

Note: For the transformation shown in Figure 1, a batch of 100 transactions that satisfy rules 1–4 in the proportion 50, 25, 10, 10, respectively (and 5 for ELSE) will require a minimum 318 comparison executions; for the transformation shown in Figure 2, the transactions will require a minimum 288 comparison executions (see Table 1).

RECEIVED JULY, 1965.

A Comparison of the Primal-Simplex and Primal-Dual Algorithms for Linear Programming

R. K. MUELLER AND L. COOPER
Washington University, St. Louis, Missouri*

A statistical comparison of the primal-dual and the more commonly used primal-simplex algorithm for solving linear programming problems has been made under the assumption of starting with a full artificial basis. Under these conditions the primal-dual method shows a statistically significant superiority on randomly generated problems. It has also been found, via a regression analysis, that the relevant parameters in determining the difference in the number of iterations between the algorithms is not only the number of constraints and the number of variables but also the ratio of the latter to the former.

1. Introduction

The simplex method for linear programming refers to an algorithm, developed in 1947 by George Dantzig, which is now widely used to solve linear programming problems. A possible inefficiency in the simplex method occurs if a large number of artificial vectors must first be added to the problem in order to obtain an initial basic feasible solution, that is, one which has non-negative components. To obtain an initial basic feasible solution to the original problem, that is, the problem without artificial vectors, a special procedure is followed. This procedure, called phase I, proceeds toward a basic feasible solution to the original problem without attempting to optimize the objective function. Thus phase I could end with the solution being far from optimal [1].

Another method, proposed jointly by Dantzig, Ford and Fulkerson, but which has not been extensively explored or applied, uses duality theory to proceed toward optimality as well as feasibility in phase I. Using this algorithm, more restrictions are placed on vectors allowed to enter the basis, while still using the basic simplex method of iteration. The restrictions are chosen in such a way that when phase I is ended, the solution is optimal. This eliminates the need for phase II calculations.

One would naturally wish to know whether the primal-dual method is indeed superior to the primal-simplex method. Also one might wish to know how to go about

The computer time on the IBM 7072 at the Washington University Computer Center was made available under NSF Grant G-22296.

* School of Engineering and Applied Science.

exhibiting the truth or falsity of this conjecture. First it is necessary to define more precisely what is meant by superior. In this context superior probably means "faster;" that is, primal-dual would be judged superior to primal-simplex if it would solve comparable problems in less time on a high speed computer. "Comparable" in the context of this paper, since it is the only case that has been investigated, is a problem in which one adds a full set of artificial vectors in order to employ an identity matrix as the basis for the expression of the initial basic feasible solution.

It was observed previously that the primal-dual and primal-simplex algorithms both use the simplex method to iterate. The primal-dual however requires an additional set of calculations per iteration. If the times per iteration could in some way be shown to be about the same for both methods, the problem is then reduced to one of comparing the iteration counts of comparable problems.

Richard Mills [2] has shown that the iteration time for the primal-dual algorithm is no more than five percent longer than the time for the simplex method. With this information all that is necessary to judge the superior method is to compare iteration counts of the two methods for a representative sample of problems.

The problem of finding a large number of problems to use for comparison purposes was solved by generating the required arrays randomly.

2. Theoretical Background

The general linear programming problem may be stated: Max $z = cx$, where c is a n-component row, cost vector and x is a n-component column, solution vector. The solution vector is also subject to the constraints: $Ax = b$, $(x \geq 0)$ where A is an $m \times n$ coefficient matrix and b is the m-component requirements, column vector. For a detailed discussion of primal-simplex algorithm theory and primal-dual algorithm theory, the reader is referred to [1].

For the purposes of this paper, it is necessary to characterize certain aspects of the assumptions and variations of the algorithms used, since the terms "primal-simplex" and "primal-dual" are not unambiguous descriptions of a unique sequence of computational steps.

The primal-simplex algorithm described here is a two-phase revised simplex method which utilizes the product form of the inverse in transformation of the basis. The initial basis is a full $m \times m$ artificial basis with which phase I is begun.

Earlier work [4] tends to indicate that the use of other procedures using some legitimate vectors and some artificial vectors might reduce the number of iterations in phase I. No comparisons of these procedures with the primal-dual algorithm have yet been made, although it seems unlikely that the primal-dual algorithm as originally described will be superior to these procedures. It is possibly true, however, that modifications of primal-dual procedures could be found to reduce the number of iterations, if attention were given to this problem. Since two-phase primal simplex differs from the primal-dual algorithm in that the primal-dual method consists of *only* a phase I in which the full artificial basis is used, it was decided to compare this procedure against the corresponding primal-simplex procedure. To date, only the paper by Wolfe and Cutler [3], which gives results of comparisons of methods to reduce the number of iterations in phase I of primal simplex, is available. No similar attempts have been made to modify in any way the primal-dual algorithm. Hence, it was decided to compare the two algorithms as indicated.

The primal-dual algorithm that was employed is, as described in [1], a method for transforming the basis which is essentially the product form of the inverse technique. Because of this, the computation time per iteration was very close for both primal-simplex and primal-dual methods. This is discussed in more detail in Section 4.

It is recalled that the primal-dual algorithm is a phase I calculation which is carried out in such a way, namely, by satisfying the principle of complementary slackness at each iteration, that when a first feasible solution is obtained at the conclusion of phase I, the solution is also optimal. Hence, there is no phase II in the primal-dual algorithm. It is possible that variations in the starting basis for the primal-dual algorithm can be found similar to the preliminary experiments reported in [4]; this is presently under consideration. However, in this paper we have compared the initial basis for both algorithms as a full artificial basis.

The only earlier comparison of the two algorithms [2] was inconclusive because (1) the author failed to derive certain conclusions from his timing comparisons, and (2) he did not attempt to study a significant number of problems.

A further point concerning this study is the choice of rule for determining the vector to enter the basis. Several studies of this problem [4, 5] have been made for the primal-simplex methods. We have employed the most commonly used rule (the most negative $z_j - c_j$ in a maximization problem). Other rules would possibly decrease the number of iterations. However, variations of these rules or the rules themselves could also possibly be used to decrease the number of iterations in the primal-dual algorithm. No studies of this have been reported; this is certainly a matter that could and should be investigated.

3. Random Generation of Data

To compare the two algorithms it is necessary to compare them on some "representative" problems. The simplest procedure that could be followed, it was decided, was to generate the coefficients of A, b, c randomly for a set of problems of varying size (rows and columns) and analyze the results using standard statistical tests of significance. A computer code was written to do this.

The number of constraints, the number of variables,

and the number of nonzero elements in the coefficient matrix are input parameters in the computer code. Using this information the coefficient matrix is calculated by assigning a random number between zero and one to certain positions in the array. The remaining positions in the array are zero. The pairs of indices for the nonzero elements are chosen by assigning to each row index, starting with row one and continuing through row m in cyclic fashion, a column index, which is chosen at random from the range, one to n.

The cost coefficients are random numbers in the range, zero to one. To prevent unbounded solutions from occurring, the sum of the variables are constrained to be less than or equal to 10^5.

To insure that a feasible solution to the generated problem exists, the requirements vector is calculated as follows: A solution vector is generated randomly with components ranging from one to fifty. A matrix multiplication, $Ax = b$, is performed to calculate b.

This method of generation insures that a feasible solution to the problem exists, since one feasible solution is the one generated randomly.

4. Results

Thirty-one randomly generated linear programming problems were run to compare iteration counts of primal-simplex and primal-dual. Each problem was run with the same specifications in the primal-simplex and primal-dual problems. In other words, the same array size and density for the coefficient matrix were used. The components of the arrays were generated randomly so as to prevent bias, as much as possible, in either method. For the most part the density of the coefficient array was maintained at five percent. The number of constraints varied from 10 to 40, and the number of variables ranged from 15 to 100. The results obtained, as well as some calculated results to be discussed shortly, are tabulated in Table 1.

TABLE 1. TABLE OF RESULTS

Percent Density	*Number of Constraints*	*Number of Variables*	*Iteration Primal-Dual*	*Counts Simplex*	*Weighted Differences*	*Rank*
10.	10.	15.	10.	11.	0.5	3
20.	10.	15.	12.	16.	3.4	11
5.	14.	20.	11.	12.	0.45	2
12.	15.	20.	17.	20.	2.15	8
5.	15.	40.	16.	25.	8.2	22
5.	15.	50.	17.	23.	5.15	18
5.	15.	60.	18.	24.	5.1	16.5
5.	15.	80.	16.	28.	11.2	28
5.	15.	100.	18.	24.	5.1	16.5
5.	20.	40.	21.	27.	4.95	15
5.	20.	50.	22.	31.	7.9	21
5.	20.	60.	21.	28.	5.95	19
5.	20.	70.	22.	34.	10.9	27
5.	20.	100.	24.	41.	15.8	30
4.5	21.	22.	16.	16.	−0.8	5
5.	25.	40.	26.	31.	3.7	12
5.	25.	45.	27.	37.	8.65	23.5
5.	25.	50.	28.	42.	12.6	29
5.	25.	55.	27.	37.	8.65	23.5
5.	25.	60.	28.	39.	9.6	26
4.5	29.	35.	27.	27.	−1.35	7
7.6	30.	35.	33.	32.	−2.65	10
5.	30.	40.	31.	32.	−0.55	4
5.	30.	50.	38.	39.	−0.9	6
5.	30.	60.	33.	44.	9.35	25
5.	35.	40.	35.	37.	0.25	1
6.4	35.	40.	42.	40.	−4.1	14
5.	35.	50.	39.	45.	4.05	13
5.	35.	60.	39.	57.	16.05	31
5.	40.	50.	45.	45.	−2.25	9
5.	40.	60.	52.	61.	6.4	20

In June, 1960, Mr. Richard Mills [2] studied the iteration times for an iteration of primal-dual versus primal-simplex. Mr. Mills first discusses his computational procedure, using some techniques which he develops for saving some calculations in iterating with the primal-dual method. Using the execution time on the computer for each instruction in the iteration procedure, he is able to estimate the total time per iteration for each method. In these computations, he assumes that the average number of nonartificial vectors in the basis for both methods is one half the number of constraints. Also he assumes only one vector is added to the basis of the restricted primal before new dual slacks are computed. The latter assumption is borne out by actual experimentation in the present study.

The results of Mr. Mills' calculations, where m and n are parameters corresponding respectively to the number of constraints and variables are:

(1) Primal-dual iteration time in machine cycles:

$$77m^2 + 374.5m + 30mn + 96n + 952$$

(2) Simplex iteration time in machine cycles:

$$77m^2 + 244.5m + 30mn + 80n + 340.$$

The advantage of simplex over primal-dual per iteration in machine cycles is $130m + 16n + 612$.

For $m = 20$ and $n = 100$, Mr. Mills ran a test problem and timed the iterations. He found that the primal-dual iteration times were about five percent longer than the simplex iteration time, which agrees with the theoretical predictions using the above relations [2].

Using Mr. Mills' theoretical results, some interesting further results can be deduced. Suppose first the worst case is examined; that is, in the situation where the number of variables is approximately equal to the number of constraints, the ratio of the time advantage of simplex to the iteration time of simplex is

$$\frac{146m + 612}{107m^2 + 324.5m + 340}.$$

In this case, where the ratio is largest for a given number of constraints, the ratio is less than .05 for m greater than or equal to 28.

A more likely situation is one in which there are about twice as many variables as constraints. The ratio of the

ime advantage of simplex to the iteration time of simplex ecomes

$$\frac{162m + 612}{137m^2 + 404.5m + 340}.$$

f m is greater than 24.5, this ratio is also less than .05. Thus it is a fair assumption in practically all situations hat the time difference in the two iterations is no larger han five percent.

If now the primal-dual iteration counts are weighted by .05, the bias for the iteration time difference is taken into ccount. When this weighted count is subtracted from he simplex-iteration count for the data in Table 1, the esults are tabulated in the *Weighted Differences* column f that table.

A method of testing whether or not the population nedian, for the weighted iteration difference, is zero is alled the signed-rank test. This test is the nonparametric nalogue of the parametric t-test; that is, here no assumption is made about the underlying statistical population of differences. The computational procedure is, first, to eparate the differences into two classes: those that are egative and those that are positive. The sum of the ranks f positive and negative differences are computed, and he absolute value of these sums is compared. The absolute alue of the smaller of the two sums is designated T. The test statistic for N (the sample size) greater than hirty is given by:

$$z = \frac{2T + 1 - N(N + 1)/2}{(N(N + 1)(2N + 1)/6)^{1/2}}$$

where z is a normal variate [3].

For the data in Table 1, T is 55 which corresponds to he sum of the ranks of negative differences; the value of V is 31. Thus z is computed as:

$$z = \frac{110 + 1 - 31(32)/2}{((31)(32)(63)/6)^{1/2}} = -3.76.$$

This value is highly significant. We use significance here n its statistical sense; i.e., the probability that this differnce in weighted iteration count is due to chance is vanishingly small. One can therefore conclude that the eighted iteration count of primal-dual is significantly ss than that of simplex.

It is evident from an examination of Table 1 that there s a consistently high positive difference in the iteration ounts when the ratio of the number of variables to contraints is about two or more. However, it would be deirable to state more precisely under what conditions the rimal-dual is superior to simplex. Also one would like to now what variables affect the difference in iteration ount and in what way the significant variables affect the esult.

A regression analysis seems to be a simple logical approach to this problem. It is not obvious, at first, what ort of regression model to choose or what independent ariables to use along with the dependent variable, the iteration difference. After experimenting with several linear regression models in terms of the variables m, n and n/m, a model was chosen which is the simplest in form with a relatively high squared correlation coefficient, R^2. The resulting regression equation is:

$$d_i = 17.38 + .7709n - 1.064m - 10.96n/m$$

with $R^2 = .7185$. Each of the regression coefficients is significantly different from zero. The t values are:

$$t = 6.5544 \quad \text{for} \quad B_1 = .7709$$

$$t = 5.7377 \quad \text{for} \quad B_2 = -1.064$$

$$t = 5.2979 \quad \text{for} \quad B_3 = -10.976$$

An F-test was also run to test the hypothesis that the total relation is zero. The computed F with 3 and 27 degrees of freedom is:

$$F = \frac{27R^2}{2(1 - R^2)} = 22.97,$$

which is highly significant as compared to the tabular value of 2.96. Again, this result indicates that the probability that there is no relation of the type indicated by the equation, i.e., that this relationship is due to chance, is vanishingly small. Indeed, it was outside the range of any tables of the F-distribution that we could find.

It should be noted that as n increases while m and n/m remain nearly constant, the iteration difference is increased. On the other hand, increasing m, while the other independent variables remain nearly fixed, decreases the iteration difference, d_i, as expected.

Notice however that there is not just a linear relation involving m and n here, but there is also a linear term in the variable n/m. When n/m becomes greater than or equal to two, the constant term no longer cancels its effect, so for example the iteration difference is not always increased by increasing n.

5. Conclusions

Under the assumptions made in this paper, namely, that both methods begin with a full artificial basis, the results of the data analysis indicate a definite superiority of the primal-dual algorithm over primal-simplex. If modifications of this assumption are made for either method, this conclusion may have to be modified.

By using the computational techniques suggested by Mr. Mills, the times per iteration are less than five percent different for problems with 28 or more constraints. The results in Table 1 indicate a smaller iteration count per problem for primal-dual over simplex. Combining these results, the conclusion is that the primal-dual algorithm is indeed faster than simplex, for solving linear programming problems, if a phase I with a full artificial basis is employed. Even if the worst possible situation prevailed, where the number of variables equals the number of constraints so that primal simplex cannot help but proceed

toward optimality as well as feasibility in phase I, the primal-dual is still about as good as primal simplex. This is an extremely important finding. In effect, it says that it is never undesirable to use primal-dual instead of primal-simplex. On the other hand, when the ratio of n to m is about two or more, it becomes extremely undesirable to use simplex rather than primal-dual.

Finally the results of the regression analysis indicate that the important independent variables affecting the size of the iteration difference are: the number of constraints, the number of variables, and the ratio of the number of variables to constraints.

RECEIVED JUNE, 1965

REFERENCES

1. HADLEY, G. *Linear Programming*. Ch. 3-8. Addison-Wesley Publishing Co. Inc., Reading, Mass., 1963.
2. MILLS, R. An evaluation of the primal-dual algorithm for linear programming. Masters Thesis, Massachusetts Institute of Technology, Cambridge, Mass., June 1960.
3. CLELLAND, R. C., AND TATE, M. W. *Nonparametric and Shortcut Statistics in the Social, Biological, and Medical Sciences*. Interstate Printers and Publishers, Inc., Danville, Ill., 1957.
4. WOLFE, P., AND CUTLER, L. Experiments in linear programming. In *Recent Advances in Mathematical Programming*, R. L. Graves, and P. Wolfe (Eds.), McGraw-Hill Book Co., New York, 1963.
5. KUHN, H., AND QUANDT, R. E. An experimental study of the simplex method. Proc. Symp. in Applied Mathematics, Vol. XV, Amer. Math. Soc., Providence, R. I., 1963.

NAUR—continued from page 676

The Integration Program

```
begin integer k, n; real u, w;
real procedure Romberg (a, b, x, f, delta, max ord);
value a, b, delta; real a, b, x, f, delta; integer max ord;
comment Romberg integrates the function f(x) from a to b.
  The parameters are:
    a:        Lower limit
    b:        Upper limit
    x:        The real variable of integration
    f:        The function to be integrated, given as a real
                depending on x
    delta:    A relative tolerance
    max ord:  On entry the maximum number of iterations. On
                exit the number of iterations performed;
  begin real step, I1, I2, sum, error, f0;
  integer k, p, j;
  array trapez[1: max ord+1];
  step := b - a;
  x := a; I1 := f;
  x := b;
  trapez[1] := (I1 + f) × step/2;
  I1 := 0;
  for k := 1 step 1 until max ord do
    begin
    sum := error := 0;
    step := step/2; p := 2↑k;
  for j := p -1 step -2 until 1 do
    begin
    x := j/p;
    x := x × a + (1-x) × b; f0 := f;
    I2 := sum + f0;
    error := error + (if abs(f0) > abs(sum)
      then sum - (I2-f0) else f0 - (I2-sum));
    sum := I2;
    end summation of function values and errors;
  I2 := trapez[k+1] := trapez[k]/2 + (I2+error) × step;
  comment: This was a trapezoidal integration with 2↑k
    subintervals;
  p := 1;
  for j := k step -1 until 1 do
    begin p := 4 × p;
    I2 := trapez[j] := (I2×p-trapez[j])/(p-1)
      end the extrapolation from the k first approximants;
    if abs(I2-I1) ≤ delta × abs(I2) then go to finis;
    I1 := I2
    end for k, i.e., a whole iteration;
finis:
    max ord := k;
    comment: For the effect of this statement, see Section 2;
    Romberg := I2
end Romberg procedure;

integer d;
d := drumplace; gierdrum(33, w);
  begin Boolean array T[1: w]; integer p;
  writetext((<
Sæt KB)); typechar; drumplace := d; from drum(T);
for p := 0 step 1 until 25 do
  begin array A[0:41×p]; real B;
if kbon then gierproc(T[3]) else typechar;
k := 5; n := 7;
B := Romberg(0, 1, u, if u=0∨u=1 then 0 else
      Romberg(0, exp(-k/sqrt(-ln(u))), w,
  if w = 0 then 0 else (-ln(w))↑(-n), 0.01, 12)/(ln(u))↑6, 0.01,
    12);
if kbon then begin gierproc(T[2]); output((ndd), p) end
    else begin writecr; write((d.dd10-dd), B);
      write((-nddd.d), p, typein)
      end;
  end
  end
end;
```

RECEIVED JUNE, 1964

REFERENCES

1. NAUR, P. The design of the GIER ALGOL compiler. *BIT 3* (1963), 124-139, 145-166.
2. NAUR, P. (Ed.) A manual of GIER ALGOL III. Regnecentralen, Copenhagen, 1964.
3. SCHWARZ, H. R. An introduction to ALGOL 60. *Comm. ACM 5* (Feb. 1962), 82-95; see especially p. 94.
4. GRAM, C. Definite integral by Romberg's method. *BIT 4* (1964), 54-60.