

T-1725

SURROGATED LINEAR PROGRAMMING

By

Thomas J. Holloran

ARTHUR LAKES LIBRARY
COLORADO SCHOOL of MINES
GOLDEN, COLORADO 80401

11025/64

ProQuest Number: 10781956

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10781956

Published by ProQuest LLC (2018). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

A Thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science.

Signed: Thomas J. Kellomaki
Student

Golden, Colorado

Date: 4/22, 1975

Approved: [Signature]
Thesis Advisor

[Signature]
Head of Department

Golden, Colorado

Date: April 22, 1975

ARTHUR LAKES LIBRARY
COLORADO SCHOOL of MINES
GOLDEN, COLORADO 80401

ABSTRACT

It is not uncommon for many firms in the mineral industry to run large linear programming codes optimizing refinery production or transportation scheduling which may take 30 hours or more of computer run time. Therefore, they are always interested in new techniques which may reduce this computation time. Surrogated linear programming is a non-simplex type of linear programming which has been promoted to be from two to ten times computationally faster than the traditional simplex method.

The purpose of this paper is three-fold: to develop and discuss a simple iterative technique for use in the surrogated linear programming algorithm; to discount allegations about the computational efficiency of surrogated linear programming; and to propose a method of combining surrogated linear programming with the simplex method.

Some published example problems are solved to emphasize my conjectures and some suggestions for further study are made.

ARTHUR LAKES LIBRARY
COLORADO SCHOOL of MINES
GOLDEN, COLORADO 80401

TABLE OF CONTENTS

Introduction	1
The Surrogated Linear Programming	
Algorithm	5
A Simple Method of Changing Surrogate	
Multipliers	9
Convergence Limitations of Surrogated	
Linear Programming	15
Combining Surrogation with Simplex	20
Conclusions and Suggestions for	
Further Study	25
Bibliography	27
Appendix	28

LIST OF TABLES AND FIGURES

Table 1.	Published Results of Dittman and Staats	4
Table 2.	Solution of a One Constraint LP Problem	8
Table 3.	Comparative Solution of Example Problems	18
Table 4.	Surrogated LP Implications VS. Example Problem Optimal Results	23

ACKNOWLEDGMENTS

I would like to thank Dr. R.E.D. Woolsey for his educational assistance and inspiration.

I would also like to thank the Colorado School of Mines and the State of Colorado for their financial assistance while I was a graduate student at the Colorado School of Mines.

ARTHUR LAKES LIBRARY
COLORADO SCHOOL of MINES
GOLDEN, COLORADO 80401

INTRODUCTION

Linear programming (LP) has gained widespread application in industries throughout the world since the development of the simplex algorithm by George Dantzig in 1947. Some of the applications of linear programming in the mineral industry are described by Symonds(1955) and Henderson(1958).

In the real world, linear programming problems are always very large and often require immense amounts of computer time to execute. Although recent developments have increased linear programming efficiency somewhat, Smart (1960) estimated solution time on a UTECOM computer - equivalent to an IBM 701 - to be $20 \times n \times m$ milliseconds per pivot where n is the number of problem variables and m is the number of problem constraints. A small refinery problem may have 500 variables and 200 constraints or have an estimated solution time of $20 \times 500 \times 200/1000 = 2000$ seconds or over 30 minutes.

Therefore, suggestions for methods of reducing linear programming computation times are always welcomed. John Dittman and Glenn Staats(1973b,p.327-332) recently published the results in Table 1 obtained using a surrogated linear programming algorithm. These results imply considerable time savings by this non-simplex technique and have promoted this study.

Surrogated linear programming solves a series of one constraint linear programming problems. The one constraint (surrogate constraint) is formed by weighting and summing all of the original problem constraints in such a way that a solution closer to optimal (infeasible, better than optimal) is obtained at each stage. Fred Glover (1965) first introduced the idea of surrogation as an aid in solving integer programming problems.

The most important step in the surrogated LP algorithm is changing the constraint weights (surrogate multipliers) to obtain the better surrogate solution at each stage. I discovered a relatively simple method of changing weights which I develop and discuss in this paper. Basically it consists of plugging the surrogate solution back into the original constraints and then multiplying these constraint values by the old constraint weights to obtain the new weights.

There are some problems in getting the surrogated linear programming algorithm to converge to the optimal solution. No proof of convergence has been attempted and neither Dittman (1973a) nor Dittman and Staats (1973b) supplies all of the "tricks" they used to induce convergence and obtain the results in Table 1.

Because of these convergence problems, the surrogated LP algorithm may not prove to be a very good alternative to the simplex method. Fred Glover has suggested that

surrogation may possibly be used to merely aid the simplex method by determining which variables will be basic at optimality and by determining tight and loose constraints.

This study expands on the above topics and solves some example problems to illustrate the described methods and support my opinions. Areas for further study of surrogation in linear programming are also suggested.

ARTHUR LAKES LIBRARY
COLORADO SCHOOL of MINES
GOLDEN, COLORADO 80401

Table 1. Published Results of Dittman and Staats

<u>PROBLEM</u>	<u>SIZE</u>	<u>SIMPLEX MPS TIME CPU SECS</u>	<u>SURROGATED TIME CPU SECS</u>	<u>RATIO OF MPS TIME</u>
Example 1	2 X 9	4.56	1.28	3.57
Example 2	3 X 14	3.804	2.64	1.44
Example 3	6 X 2	3.23	0.28	11.51
Example 4	10 X 2	3.636	0.76	4.77
Example 5	7 X 4	3.606	1.74	2.02
Example 6	5 X 4	2.40	0.93	2.58
Example 7	9 X 7	4.068	2.77	1.47
Example 8	5 X 6	2.70	0.78	3.46
Example 9	2 X 2	2.886	0.46	5.88
Example 10	2 X 2	2.268	0.71	3.20
Example 11	2 X 2	3.000	0.88	3.41
Example 12	6 X 6	3.258	2.06	1.58
Example 13	4 X 4	2.814	0.66	4.26
Example 14	30 X 30	16.716	7.56	2.20

THE SURROGATED LINEAR PROGRAMMING ALGORITHM

Surrogated linear programming solves linear programming problems by combining all of the linear constraints into a single constraint and solving a series of one constraint linear programming problems.

Note that any n variable, m constraint linear programming problem can be written in the following form:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^n c_j x_j \\ \text{subject to} \quad & \sum_{j=1}^n a_{ij} x_j \leq b_i \quad \text{for } i=1, \dots, m \\ & x_j \geq 0 \quad \text{for } j=1, \dots, n \end{aligned}$$

where $b_i = 1, -1$, or 0 .

The surrogate vector is defined to be the vector of normalized constraint multipliers $\underline{h} = (h_1, h_2, \dots, h_m)^T$ such that $\sum_{i=1}^m h_i = 1$ and $0 \leq h_i \leq 1$ for $i=1, \dots, m$. When the original LP constraints above are multiplied by this surrogate vector and summed, the following one constraint LP problem results:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^n c_j x_j \\ \text{subject to} \quad & \sum_{j=1}^n s_j x_j \leq b_s \\ & x_j \geq 0 \quad \text{for } j=1, \dots, n \end{aligned}$$

where $s_j = \sum_{i=1}^m h_i a_{ij}$ and $b_s = \sum_{i=1}^m h_i b_i$.

ARTHUR LAKES LIBRARY
 COLORADO SCHOOL OF MINES
 GOLDEN, COLORADO 80401

The solution to the above one constraint LP problem is trivial (See Table 2). Hence, the problem becomes that of finding the normalized surrogate vector $\underline{h} = (h_1, h_2, \dots, h_m)^T$ such that the solution to the surrogated LP problem is identical to the solution to the original LP problem. Dittman (1973a, p.22-26) discusses the existence and uniqueness of this optimal surrogate vector \underline{h}^* in depth.

The basic algorithm for finding \underline{h}^* is as follows:

- Step 1: Select an initial surrogate vector $\underline{h} = (h_1, h_2, \dots, h_m)$ such that $\sum_{i=1}^m h_i = 1$ and $0 \leq h_i \leq 1$ for $i=1, \dots, m$ which has a bounded surrogate solution found from Table 2.
- Step 2: Plug the surrogated linear programming (SLP) solution (only one variable, x_j^* , will have a value other than zero) back into the original LP constraints and calculate the left hand sides - $LHS_i = a_{ij}x_j^*$ for $i=1, \dots, m$.
- Step 3: If the LP constraint i is undersatisfied, it has been given too much weight in the surrogated constraint and its multiplier h_i should be reduced. Likewise, if the i^{th} LP constraint is violated, the constraint needs to be made more influential in the SLP problem and its corresponding multiplier should be increased. Exactly satisfied constraints' multipliers remain unchanged. The surrogate vector \underline{h} must remain normalized.

Step 4: Calculate the new SLP problem solution using Table 2.

If the new set of multipliers results in a better solution (nearer optimal) and is not within the stopping criterion, go to Step 2. If a worse solution is obtained, the multipliers have been changed too much; so return to Step 3 and reduce the amount of change. If the new better solution is within some tolerance of the previous best, the optimal surrogate vector \underline{h}^* has been found.

Stop.

When the optimal surrogate vector \underline{h}^* is found by the above algorithm, the objective function value of the surrogated problem (Z_s) is equal to that of the original problem at optimality. However, the optimal problem variable values are undetermined. Dittman (1973a, p.111-118) showed that for any feasible vector of surrogate multipliers a feasible solution to the problem's dual having the same objective function value can be obtained by

$$w_i = Z_s \cdot h_i / b_s \quad \text{for } i=1, \dots, m$$

where w_i represents the i^{th} dual variable.

Techniques of changing the surrogate multipliers (Step 3 above) are discussed in more detail in the ensuing material along with the algorithm's drawbacks.

ARTHUR LAKES LIBRARY
COLORADO SCHOOL of MINES
GOLDEN, COLORADO 80401

Table 2. Solution of a One Constraint LP Problem

$c_j, j=1, \dots, n$	$s_j, j=1, \dots, n$	b_s	SOLUTION
all nonnegative	all negative or some positive some negative	Any Sign	Unbounded
some positive some negative	all negative	Any Sign	
	some positive some negative	Any Sign	Unbounded unless $c_j > 0$ implies $s_j > 0$ and $\max_j \{c_j/s_j; c_j, s_j > 0\} \leq \min_j \{c_j/s_j; c_j, s_j < 0\}$
		Positive or Zero	If bounded, $x_i^* = \begin{cases} b_s/s_k & i=k \\ 0 & i \neq k \end{cases}$ where $c_k/s_k = \max_j \{c_j/s_j; c_j, s_j > 0\}$
all nonnegative or some positive some negative	all positive	Positive	If bounded, $x_i^* = \begin{cases} b_s/s_k & i=k \\ 0 & i \neq k \end{cases}$ where $c_k/s_k = \max_j \{c_j/s_j; c_j > 0\}$
		Negative	If bounded, $x_i^* = \begin{cases} b_s/s_k & i=k \\ 0 & i \neq k \end{cases}$ where $c_k/s_k = \min_j \{c_j/s_j; c_j, s_j < 0\}$
all nonpositive	all negative or some positive some negative	Negative	$x_i^* = \begin{cases} b_s/s_k & i=k \\ 0 & i \neq k \end{cases}$ where $c_k/s_k = \min_j \{c_j/s_j; s_j < 0\}$
		Zero	$\underline{x}^* = 0$
	any sign combination	Positive	
any sign combination	all positive	Zero	No Feasible Solution
		Negative	

A SIMPLE METHOD OF CHANGING SURROGATE MULTIPLIERS

Given a feasible surrogate vector \underline{h} and a SLP problem solution x_j^* , I suggest the following method of changing surrogate multipliers:

$$h_i' = h_i \cdot \text{LHS}_i \quad \text{for } i=1, \dots, m$$

where LHS_i 's are calculated by plugging x_j^* into the original LP problem reformulated as:

$$\begin{aligned} &\text{maximize} && \sum_{j=1}^n c_j x_j \\ &\text{subject to} && \sum_{j=1}^n a_{ij} x_j + 1 - b_i \leq 1 \quad \text{for } i=1, \dots, m. \end{aligned}$$

Note that if the LHS_i is less than 1, the i th constraint is undersatisfied and its surrogate multiplier should be reduced - as is the case above when h_i is multiplied by a fraction less than 1. When LHS_i is equal to 1, the LP problem constraint is exactly satisfied and $h_i' = h_i \cdot 1 = h_i$. For violated constraints, LHS_i 's will be greater than 1 and the corresponding surrogate multipliers will be increased.

Theorem: If \underline{h}' is calculated by the above method, it will be a normalized surrogate vector.

Proof. Recall the SLP problem definition:

$$\begin{aligned} &\text{maximize} && \sum_{j=1}^n c_j x_j \\ &\text{subject to} && \sum_{j=1}^n s_j x_j \leq b_s \\ &&& x_j \geq 0 \quad \text{for } j=1, \dots, n \\ &\text{where } s_j = && \sum_{i=1}^m h_i a_{ij} \quad \text{and} \quad b_s = \sum_{i=1}^m h_i b_i. \end{aligned}$$

The SLP problem solution is $x_j^* = b_s/s_j$ where j is the index of the only nonzero variable.

Now, from the above LP problem reformulation note that

$$\text{LHS}_i = (x_j^* a_{ij} + K - b_i)/K \quad \text{for } i=1, \dots, m$$

where $K=1$.

The surrogate vector \underline{h}' is then defined by:

$$h_i' = h_i \cdot \text{LHS}_i = h_i \cdot (x_j^* a_{ij} + K - b_i)/K \quad \text{for } i=1, \dots, m.$$

Summing the surrogate multipliers of \underline{h}' :

$$h_1' + h_2' + \dots + h_m' = h_1 \text{LHS}_1 + h_2 \text{LHS}_2 + \dots + h_m \text{LHS}_m$$

$$= 1/K [h_1(x_j^* a_{1j} + K - b_1) + h_2(x_j^* a_{2j} + K - b_2) + \dots + h_m(x_j^* a_{mj} + K - b_m)]$$

$$= 1/K [x_j^* (h_1 a_{1j} + h_2 a_{2j} + \dots + h_m a_{mj}) + K(h_1 + h_2 + \dots + h_m) - (h_1 b_1 + h_2 b_2 + \dots + h_m b_m)]$$

$$= 1/K [x_j^* s_j + K - b_s]$$

$$= 1/K [(b_s/s_j) \cdot s_j + K - b_s]$$

$$= 1/K [b_s + K - b_s]$$

$$= K/K$$

$$= 1.$$

Hence, the calculated surrogate vector \underline{h}' is indeed normalized and the proof is complete.

Note that in the above theorem \underline{h} remained normalized independent of the quantity K . Therefore, K can be increased if negative LHS_1 's are incurred. Also, larger values of K result in smaller degrees of change to the surrogate multipliers. Thus, K can merely be increased to calculate LHS_1 's if the previous amount of change was too great (resulted in a worse objective function value or an unbounded solution).

Also note that the surrogate multipliers will remain normalized independent of their original sum.

An example problem is solved below to demonstrate the above described method.

$$\begin{aligned} \text{Maximize} \quad & x_1 + 2x_2 \\ \text{subject to} \quad & 3x_1 + 4x_2 \leq 1 \\ & 4x_1 + 2x_2 \leq 1. \end{aligned}$$

ARTHUR LAKES LIBRARY
COLORADO SCHOOL of MINES
GOLDEN, COLORADO 80401

***** ITERATION 1 *****

Choose an initial set of surrogate multipliers as $\underline{h} = (1, 1)$.
The surrogated problem then becomes:

$$\begin{aligned} \text{maximize} \quad & x_1 + 2x_2 \\ \text{subject to} \quad & 7x_1 + 6x_2 \leq 2. \end{aligned}$$

The surrogated solution is: $\underline{x} = (0, .333)$ and $Z_g = .667$.
Changing the surrogate multipliers:

$$\begin{aligned} LHS_1 &= 3(0) + 4(.333) = 1.333 \\ LHS_2 &= 4(0) + 2(.333) = .667 \\ h_1^1 &= h_1 \cdot LHS_1 = (1)1.333 = 1.333 \\ h_2^1 &= h_2 \cdot LHS_2 = (1).667 = .667. \end{aligned}$$

***** ITERATION 2 *****

The new set of surrogate multipliers: $\underline{h} = (1.333, .667)$.

The surrogated problem:

$$\begin{aligned} &\text{maximize} && x_1 + 2x_2 \\ &\text{subject to} && 6.667x_1 + 6.667x_2 \leq 2. \end{aligned}$$

The surrogated solution: $\underline{x} = (0, .3)$ and $Z_s = .6 < .667$.

Changing the surrogate multipliers:

$$\begin{aligned} \text{LHS}_1 &= 3\{0\} + 4\{.3\} = 1.2 \\ \text{LHS}_2 &= 4\{0\} + 2\{.3\} = .6 \\ h_1^1 &= h_1 \cdot \text{LHS}_1 = (1.333)1.2 = 1.6 \\ h_2^1 &= h_2 \cdot \text{LHS}_2 = (.667).6 = .4. \end{aligned}$$

***** ITERATION 3 *****

The new set of surrogate multipliers: $\underline{h} = (1.6, .4)$.

The surrogated problem:

$$\begin{aligned} &\text{maximize} && x_1 + 2x_2 \\ &\text{subject to} && 6.4x_1 + 7.2x_2 \leq 2. \end{aligned}$$

The surrogated solution: $\underline{x} = (0, .278)$ and $Z_s = .556 < .6$.

Changing the surrogate multipliers:

$$\begin{aligned} \text{LHS}_1 &= 3(0) + 4(.278) = 1.111 \\ \text{LHS}_2 &= 4(0) + 2(.278) = .556 \\ h_1^1 &= h_1 \cdot \text{LHS}_1 = (1.6)1.111 = 1.778 \\ h_2^1 &= h_2 \cdot \text{LHS}_2 = (.4).556 = .222. \end{aligned}$$

***** ITERATION 4 *****

The new set of surrogate multipliers: $\underline{h} = (1.778, .222)$.

Surrogate solution: $\underline{x} = (0, .265)$ and $Z_s = .529 < .556$

Calculate new surrogate multipliers:

$$\begin{aligned} h_1^1 &= h_1 \cdot \text{LHS}_1 = (1.778)1.059 = 1.882 \\ h_2^1 &= h_2 \cdot \text{LHS}_2 = (.222).529 = .118. \end{aligned}$$

***** ITERATION 5 *****

Surrogate multipliers: $\underline{h} = (1.882, .118)$.

Surrogate solution: $\underline{x} = (0, .258)$ and $Z_s = .515 < .529$

Calculate the new surrogate vector:

$$\begin{aligned} h_1^1 &= h_1 \cdot \text{LHS}_1 = (1.882)1.03 = 1.939 \\ h_2^1 &= h_2 \cdot \text{LHS}_2 = (.118).515 = .061. \end{aligned}$$

***** ITERATION 6 *****

Surrogate multipliers: $\underline{h} = (1.882, .118)$.

Surrogate solution: $\underline{x} = (0, .254)$ and $Z_s = .508 < .515$

Calculate the new surrogate vector:

$$h_1^i = h_1 \cdot \text{LHS}_1 = (1.939)1.015 = 1.969$$

$$h_2^i = h_2 \cdot \text{LHS}_2 = (.061).508 = .031.$$

***** ITERATION 7 *****

Surrogate multipliers: $\underline{h} = (1.969, .031)$.

Surrogate solution: $\underline{x} = (0, .252)$ and $Z_s = .504 < .508$.

Note that this surrogate vector produces a better value of Z_s but that it is less than 1% better than the last value obtained, which in this case is a suitable stopping criterion. Hence, this iteration establishes the optimal solution to the LP problem ($Z = .50$).

The optimal solution to the dual problem can be generated by

$$w_1 = Z_s h_1 / b_s = .50(1.969) / 2. = 0.5,$$

$$\text{and } w_2 = Z_s h_2 / b_s = .50(.061) / 2. = 0.0.$$

In this problem the optimal surrogate solution $\underline{x} = (0, .25)$ is also conveniently a feasible solution to the original LP problem. This will occur only when the original problem has only one nonzero variable in its solution.

This proposed technique of changing the surrogate multipliers has two main advantages. First, it is quick and straightforward and has many built-in characteristics that other change techniques must be modified to include.

For example, the amount of change in multipliers is automatically weighted by the size of each multiplier ($h_1' = h_1 \cdot \text{LHS}_1$) and negative left hand sides are easily taken care of by increasing the constant K. The other advantage of this proposed technique lies in the ability to manipulate this constant K. That is, K values can be kept sufficiently large to insure small but successively better surrogate solutions and previous sets of surrogate multipliers are easily generated by saving their associated constant K.

There are two main disadvantages to this technique also. The initial bounded surrogate vector must have all non-zero surrogate multipliers or the multipliers can never change from zero by the above technique. Allowing multipliers to be zero is often convenient in obtaining a bounded surrogate solution. The other disadvantage of this method is that it does not indicate a "steady state" procedure. Dittman (Appendix) bases his changes on a composite of past left hand sides which forms his "steady state" procedure and increases his rate of convergence significantly.

CONVERGENCE LIMITATIONS OF SURROGATED LINEAR PROGRAMMING

The surrogated linear programming algorithm approaches the optimal surrogate vector \underline{h}^* very rapidly with the first few changes of surrogate multipliers. However, convergence then becomes drastically slow.

One cause of this slowed convergence is obvious. As the surrogate vector \underline{h} approaches \underline{h}^* , smaller changes must be made in \underline{h} and resulting better objective function values are closer together. The stopping criterion of the surrogated algorithm helps avoid this problem but also induces error.

A major halt in the rate of convergence occurs when the surrogate multipliers are changed too much (Step 4 of the algorithm). Decreasing the amount of change needed to obtain a better solution usually takes several iterations and it is almost assured that the next change of multipliers will also be too great. By manipulating the variable K in the previously proposed change technique, it is possible to keep \underline{h} changes small enough to obtain a better solution most of the time. However, for the reasons described below, this is very seldom helpful.

The non-zero variable in the surrogate solution may alternate between several different variables and each may indicate an entirely different direction of change for \underline{h} .

Obviously a sufficiently small multiplier change can always be made that must result in a better solution. However, anything but a very small change may cause another variable to become the non-zero variable in the surrogate solution. When this occurs a worse solution will often result (the change was not in the correct direction for the new variable basis). Hence, the algorithm may converge to a suboptimal solution.

Another problem which hinders convergence is that of unboundedness. Once a bounded surrogate vector is found it is possible to change the surrogate multipliers in such a way that the new \underline{h}' will be unbounded. When this is done, \underline{h} has been changed too much and convergence is slowed as described above. It is also possible that problem coefficients can arise such that unboundedness interferes with the search for the optimal surrogate vector and a suboptimal solution occurs.

After the surrogated LP algorithm obtains its best possible solution, there are many special cases - Dittman (1973a, p.88-108) - that must be checked to insure the best possible solution has been found.

Regardless of these convergence problems, John Dittman and Glenn Staats (1973b, p.327-332) have plugged surrogated linear programming as being computationally faster than the simplex method. I programmed their algorithm (Appendix)

including some "tricks" that they don't mention but that are obviously needed. I also programmed my previously described change technique (Appendix) to iterate only while convergence is rapid. These codes, along with a simplex algorithm, were used to solve a set of test problems (Table 3).

Comparing my shortened algorithm results with those of the Dittman-Staats code shows how the curve of objective function values flattens out after the first few iterations.

None of the Dittman example problems take more than three pivots by the simplex method. Moreover, four of the sample problems take only one pivot to optimality. Dittman and Staats claim considerable time savings by surrogation over simplex in each of these problems (See Table 1). They claim surrogated LP is 3.5 times faster than simplexing in example 1, and claim that it takes 4.56 CPU seconds on an IBM 370 for one pivot on the example's 2 X 9 matrix. This computation time and simplex computation times published for the other example problems seem exceptionally large. Dittman and Staats used a large mathematical programming system package (MPS/360) to obtain the simplex solutions which may have resulted in additional loading times and/or execution times for computations unrelated to the actual simplexing.

ARTHUR LAKES LIBRARY
COLORADO SCHOOL of MINES
GOLDEN, COLORADO 80401

Table 3. Comparative Solution of Example Problems

PROBLEM SIZE	PROPOSED METHOD			SIMPLEX METHOD			DITTMAN-STAA'S METHOD		
	BEST OBJ FUNCTION VALUE	# OF 1 CONSTR LP'S	CPU TIME	OPT OBJ FUNCTION VALUE	# OF PIVOTS	CPU TIME	BEST OBJ FUNCTION VALUE	# OF 1 CONSTR LP'S	CPU TIME
Dittman-									
Example 1	.11199	17	0.26	.11111	1	0.25	.11121	30	0.33
Example 2	.067788	9	0.32	.06604	3	0.46	.067312	114	0.96
Example 5	2.8588	4	0.20	2.0000	1	0.29	2.0007	82	0.66
Example 6	7.0028	9	0.20	7.0000	2	0.32	7.0043	20	0.21
Example 11	2.0377	4	0.14	2.03448	2	0.29	2.0349	146	0.28
Example 13	.66768	16	0.20	.66667	1	0.32	.66703	20	0.22
Example 15	1.7936	4	0.18	1.78125	3	0.34	1.7879	138	0.52
Example 19	1.0025	17	0.30	1.0000	2	0.29	1.0010	24	0.36
Example 20	.29734	7	0.22	.25620	3	0.27	.26518	119	0.55
Symonds-									
Example 1	3313.5	4	0.17	2500.	3	0.14	3146.7	17	0.36
Example 2	92265.	22	0.42	67833.	5	0.28	79834.	200	1.08
Example 3	11410.	6	0.33	7750.0	8	0.48	9896.0	196	1.69
Hillier-Lieberman-									
Example 1	539.90	11	0.19	525.00	3	0.19	534.25	229	0.63
Example 2	28734.	25	0.58	7166.7	8	0.56	17554.	251	3.71
Example 3	643400.	2	0.36	342500.	9	0.73	356870.	204	2.40
Danø-	348650.	53	0.86	214368.	5	0.57	409900.	157	1.94
Charnes & Cooper-									
Example 1	23316000.	53	1.81	7264489.	9	0.91	21397000.	178	4.69
Example 2	18869.	8	0.31	15442.	8	0.31	27686.	130	1.04
Example 3	263560.	52	2.11	53955.	8	1.45	57856.	354	11.03

The small simplex LP program (Appendix) used to produce the figures in Table 3 solved the example problems of Dittman and Staats faster than the surrogated LP algorithm in most cases. Solutions to the other published sample problems in Table 3 heavily favor the simplex method for computational speed and accuracy.

COMBINING SURROGATION WITH SIMPLEX

Surrogated linear programming has been proposed as an alternative to the simplex LP algorithm. I doubt that this will ever occur. Convergence limitations and induced errors in problem solutions are severe problems of the surrogated algorithm. Even if these problems are overcome by developing or improving on iterative multiplier change techniques like the one proposed in this paper, I don't think surrogated LP can replace the simplex method. The simplex tableau has too much to offer. Sensitivity analysis, determining alternate optimal solutions, recognizing infeasible problem formulations, and primal-dual relationships are only a few areas where simplex far outdistances surrogated LP.

It might be possible to combine surrogated LP with the simplex method, using the advantages of both techniques. Surrogated LP often tends to drive the surrogate multipliers of loose or non-binding constraints to zero very quickly. Also, surrogate multipliers of the most binding constraints often become large very quickly. If a few surrogate iterations were run before starting to simplex, all of the determined loose constraints could be eliminated from the problem; which may eliminate some extreme points on the convex region to be examined by the simplex method.

Constraints determined to be tight at optimality would have slack variables with value zero. These constraints can be viewed as possible pivot rows for new basic variables. By pivoting several variables into these designated rows at once it might be possible to skip over the convex region and hence skip several extreme points the simplex method would normally examine. The variables to be pivoted into these tight constraints might be determined as those variables that have coefficients nearest to being zero in an updated surrogate objective function. That is, as each surrogate one constraint LP problem is solved, the resultant non-zero (basic) variable's coefficient in row zero (the updated objective function) is eliminated. Note that this updated objective function is in no way used in the surrogated LP process itself.

I used the program with my surrogation technique to determine the loose constraints, potential pivot rows (tight constraints), and potential pivot variables (basic variables) for the previous example problems. Results appear in Table 4. Recall that this program only iterates while convergence is rapid in the first few iterations. The initial surrogate multipliers were all 1 and a constraint was determined loose if its corresponding multiplier was less than 0.5 and tight if its corresponding multiplier was greater than 1.5. The number of potential pivot variables was determined by the number of potential pivot rows or tight constraints.

The surrogation technique very accurately picked the optimal basic variables and the binding and non-binding constraints for all of the Dittman example problems. However, it didn't do as well on the other examples. Implications of the surrogate technique can obviously be expected to be more accurate when the surrogate procedure results in a solution that is close to the optimal one. The Dittman problems all had loose constraints that were extremely loose and tight constraints that were generally binding. These types of problems are especially conducive to surrogation. Problems with many tight or nearly tight constraints are generally difficult for the surrogate algorithm since it can't decide which constraints are most binding and changes directions of convergence. The updated surrogate objective function worked well in picking the basic variables in all of the example problems.

There are some problems in combining these results with the simplex method. The major problem lies in pivoting several variables into the simplex tableau at once. It is possible to generate tableaus with negative right-hand sides (infeasible), tableaus with negative right-hand sides and negative row zero coefficients both, tableaus with all positive right-hand sides and all positive row zero coefficients with a non-optimal objective function value, etc. Some general guide rules can be set up to avoid these

Table 4. Surrogated LP Implications VS. Example Problem Optimal Results

PROBLEM	IMPLIED BY SURROGATED LP		ACTUAL AT OPTIMALITY	
	BASIC NON-BINDING VBLs CONSTRAINTS	BINDING CONSTRAINTS	BASIC VBLs CONSTRAINTS	TIGHT CONSTRAINTS
Dittman-				
Example 1	1,2	6,7	1	6,7
Example 2	1,3	8,9	1,3	8,9
Example 5	3	4	2,3	1,3,4
Example 6	3,4	1,4	3,4	1,4
Example 11	2	None	1,2	1,2
Example 13	2	2	2	2
Example 15	7	None	3,6,7	1,2,3
Example 19	1	7	1,2	3,7
Example 20	1,2	3,5	1,2	5,6
Symonds-				
Example 1	1	None	2,3	1,3
Example 2	4	None	1-5	2,3,5-7
Example 3	4	None	2-6	1,3,4,7-9
Hillier-Lieberman-				
Example 1	1	4	1,2,3	1,2,4
Example 2	3,4	3,4,7	1-3,5-7	1-3,6,7,9,10
Example 3	2,5	10	1-3,5,6	4-6,10-13
Danø-	5	1-4	1-3,5-8	4-7,10-13
Charnes and Cooper-				
Example 1	6,11	1,2,4,7-15	1,2,4-6,11,12,16	1-5,7,8,11
Example 2	6	1	1,3-8	1-3,5-8
Example 3	14	None	1-3,5,9-11,16	9,10,17,18
			19,20	21-24

multiple pivot problems but incorporating them in a simplex code is a problem in itself. Even if surrogating is instrumental in avoiding some extreme points on the convex region, the time used for surrogating, multiple pivoting, and eliminating constraints must be justified by the reduced simplex time.

CONCLUSIONS AND SUGGESTIONS FOR FURTHER STUDY

The method I have described for changing the surrogate multipliers may prove very useful. It might be applied to surrogation techniques now being used in areas of mathematical programming outside LP such as integer, geometric, or quadratic programming. It is very quick and simple for use when only a few surrogate iterations are required (determining loose constraints, combining surrogation with another technique, etc.) If convergence in solving LP problems by surrogation alone can be improved - as yet convergence by any technique cannot be proven - it may even be useful as an alternative to the simplex method.

This study shows surrogated linear programming to be inferior to the simplex method. The computation times published by Dittman are somewhat misleading and I suggest that a more representative test be made on some real problems, or at least on some real-sized problems.

Surrogated LP can sometimes tell some valuable things about the optimal LP solution. The actual efficiency of combining the surrogation and simplex methods needs to be tested by incorporating the surrogate LP algorithm in a simplex code which uses its information and comparing it to the regular simplex method.

I believe that surrogated LP will never become an efficient alternative to the simplex algorithm for solving

the general linear programming problem. Its future lies in solving special types of LP problems that are especially adaptive to surrogation, relating the surrogation results to the simplex tableau, or using surrogate results to obtain only certain information about the problem. I suggest research in all of these areas before any more study on convergence is done.

BIBLIOGRAPHY

- Charnes, A., and Cooper, W.W., 1961, Management Models and Industrial Applications of Linear Programming: New York, Wiley and Sons, 2 vols., 859 p.
- Danø, Sven, 1965, Linear Programming in Industry: New York, Springer-Verlag/Wien, 120 p.
- Dittman, John, 1973a, Surrogated Linear and Quadratic Programming: PhD Dissertation, University of Missouri, Columbia, Missouri.
- _____ and Staats, Glenn, 1973b, Surrogated Linear Programming: AIIE Transactions, v.5, no.4, p. 327-332.
- Glover, Fred, 1965, A Multiphase-Dual Algorithm for Zero-One Integer Programming Problems: Operations Research, v.13, p. 879-919.
- Henderson, James, 1958, The Efficiency of the Coal Industry: An Application of Linear Programming: Cambridge, Harvard University Press.
- Hillier, F.S., and Lieberman, G.J., 1967, Introduction to Operations Research: San Francisco, Holden-Day, Inc., 639 p.
- Smart, R.G., 1960, Computers and Linear Programming: Proceedings of a Seminar at the University of New South Wales, eds. Layton and Thornton.
- Symonds, G.H., 1955, Linear Programming: The Solution of Refinery Problems: New York, Esso Standard Oil Co., 74 p.

APPENDIX

ITERATIVE PROCEDURE OF DITTMAN AND STAATS

This procedure is based on a difference quantity d_i . When the surrogated solution (x_j^*) has been determined for some h , the variable is plugged back into the original LP constraints and left hand sides (LHS_i) are calculated as stated earlier. Then a d_i corresponding to each constraint i is determined by

$$d_i = LHS_i - b_i.$$

It can be shown that all d_i will be zero or some will be positive and some negative for each SLP solution. If d_i is negative, the constraint i is undersatisfied and its corresponding multiplier should be reduced whereas a positive d_i indicates a violated constraint and a h_i that needs to be increased.

The new surrogate vector h' is then calculated as follows:

$$h'_i = \begin{cases} h_i - d_i Q_0 / \sum_{i \in S_n} d_i & \text{for } i \in S_n \\ h_i + d_i Q_0 / \sum_{i \in S_p} d_i & \text{for } i \in S_p \end{cases}$$

where S_n denotes the set of undersatisfied or exactly satisfied constraint indices and S_p denotes the set of indices of violated constraints. Q_0 is an arbitrarily chosen quantity, but is generally $\sum_{i \in S_n} h_i / 2$.

This method insures that succeeding surrogate vectors remain normalized. If a new SLP solution is worse than one previously obtained or unbounded, Q_0 is reduced and \underline{h}' is recalculated resulting in a smaller change in the surrogate multipliers.

Note that although the above technique maintains $\sum_{i=1}^m h_i = 1$ it may be possible for some h_i' to be negative; which is not allowed. This can be remedied by reducing Q_0 , but if some h_i , $i \in S_n$ is extremely small with d_i relatively large the overall change of the surrogate vector may be sufficiently small to converge to a suboptimal solution. It is suggested that this be remedied by allowing the h_i which results in a negative h_i' to be reduced to an arbitrary positive value as $h_i/4$. This requires that Q_0 be changed to $\sum_{i \in S_n} h_i' - \sum_{i \in S_n} h_i$ when calculating h_i' for $i \in S_p$ in order that the surrogate vector \underline{h}' remains normalized.

The above change technique doesn't take into account the relative size of the surrogate multipliers. Since it is appealing to do so, the d_i quantities can be weighted by allowing

$$d_{i_{\text{new}}} = h_i d_i \quad \text{for } i=1, \dots, m.$$

The above method remains unchanged except $d_{i_{\text{new}}}$'s are used in the place of the d_i 's.

Using running sums of the calculated d_1 's at each stage is used for better convergence. When using running sums it is possible a better solution cannot be found for a given set of d_1 's. When this occurs, the last d_1 is added again to the running sum until a better solution is found.

```

C*****
C  DITTMAN - STAATS SURROGATED LINEAR PROGRAMMING METHOD
C
C*****
      DIMENSION C(50),AS(50)
      COMMON LHS(100),W,D(100),B(100),U(100,2),A(100,50)
      COMMON K,L,ISET,Z,OBJ,AVG,IMIN,WMIN,IVBL,NOADJ,NODI,
1         Q0,D1,D2,M,BS,TURNCE,ICHNGE,IUNB,SBS,INEG
      DOUBLE PRECISION FMTC(3),FMTB(3)
      REAL LHS
100     FORMAT(2G)
101     FORMAT(3A10)
102     FORMAT(55H)
C*****
C
C  ** PROBLEM **
C  MAXIMIZE:
C
C      C(1)*X(1) + C(2)*X(2) + ... + C(N)*X(N)
C  SUBJECT TO:
C      A(1,1)*X(1) + A(1,2)*X(2) + ... + A(1,N)*X(N) .LE. B(1)
C      A(2,1)*X(1) + A(2,2)*X(2) + ... + A(2,N)*X(N) .LE. B(2)
C      :
C      :
C      A(M,1)*X(1) + A(M,2)*X(2) + ... + A(M,N)*X(N) .LE. B(M)
C
C  ** DATA **
C  RECORD 1: PROBLEM IDENTIFICATION
C  RECORD 2: NUMBER OF PRIMAL VBLS, NUMBER OF PRIMAL CONSTRAINTS
C  RECORD 3: NONZERO NUMBER IF INTITAL SURROGATE VECTOR INPUT
C  RECORD 4: FORMAT FOR OBJ FUNCTION AND CONSTRAINTS
C  RECORD 5: FORMAT FOR RHS
C  RECORD 6: INITIAL SURROGATE VECTOR (ONLY IF INPUT)
C  RECORD 7: OBJ FUNCTION COEFFICIENTS C(J)'S
C  RECORD 8: RHS OF CONSTRAINTS B(I)'S
C  RECORD 9+: COEFFICIENTS OF CONSTRAINTS A(I,J)'S
C*****
      READ(5,102)
      WRITE(6,102)
      READ(5,100) N,M
      READ(5,100) INPUT
      READ(5,101) (FMTC(I),I=1,3)
      READ(5,101) (FMTB(I),I=1,3)
      IF(INPUT.NE.0) READ(5,FMTB) (U(I,1),I=1,M)
      READ(5,FMTC) (C(J),J=1,N)
      READ(5,FMTB) (B(I),I=1,M)
      DO 2 I=1,M
2     READ(5,FMTC) (A(I,J),J=1,N)
C*****

```

C FORM NORMALIZED PROBLEM AND INITIALIZE VBLS
C

```

DO 3 I=1,M
  IF(B(I).EQ.0.) GO TO 3
  DO 4 J=1,N
4    A(I,J)=A(I,J)/ABS(B(I))
    B(I)=B(I)/ABS(B(I))
3    CONTINUE
    OBJ=1.E35
    ISET=0
    NOADJ=0
    IUNB=0
    ICHNGE=0
    AVG=1.E35
    ITER=0
    K=1
    L=2

```

C*****

C INITIALIZE SURROGATE MULTIPLIERS
C

```

  IF(INPUT.NE.0) GO TO 500
  DO 5 I=1,M
  D(I)=0.
5    U(I,K)=1.

```

C*****

C FORM SURROGATE CONSTRAINT
C

```

500  IN=0
     IX=0
     BS=0.
     DO 50 J=1,N
50   AS(J)=0.
     DO 6 J=1,N
     DO 6 I=1,M
6    AS(J)=AS(J) + U(I,K)*A(I,J)
     DO 7 I=1,M
7    BS=BS + B(I)*U(I,K)

```

C*****

C SOLVE THE SURROGATED PROBLEM
C

```

  IPA=0
  INA=0
  IPC=0
  INC=0
  AMAX=0.
  AMIN=1.E35
  DO 200 J=1,N
  IF(C(J)) 221,222,222
221  INC=-1
     IF(AS(J).GE.0.) GO TO 220
     INA=-1

```

```

      IF(C(J)/AS(J).GE.AMIN) GO TO 200
      AMIN=C(J)/AS(J)
      IN=J
      GO TO 200
220   IPA=1
      GO TO 200
222   IPC=1
      IF(AS(J).LE.0.) GO TO 223
      IPA=1
      IF(C(J)/AS(J).LE.AMAX) GO TO 200
      AMAX=C(J)/AS(J)
      IX=J
200   CONTINUE
      IA=IPA + INA
      IC=IPC + INC
      IF(IC.LT.0.AND.BS.GT.0.) GO TO 14
      IF(IC.LT.0.AND.IA.LT.1.AND.BS.EQ.0.) GO TO 14
      IF(IA.GT.0.AND.BS.EQ.0.) GO TO 14
      IF(IA.GT.0.AND.BS.LT.0.) GO TO 15
      IF(AMIN.GE.AMAX) GO TO 224
223   K=3-L
      IF(ITER.EQ.0) GO TO 123
      IUNB=1
      GO TO 300
123   WRITE(6,104)
104   FORMAT(15X,'UNBOUNDED INITIAL SURROGATE VECTOR')
      STOP
14    Z=0,
      IVBL=1
      W=0,
      GO TO 225
15    WRITE(6,105)
105   FORMAT(15X,'SURROGATE PROBLEM HAS NO FEASIBLE SOLN')
      STOP
224   IVBL=IN
      IF(BS.GE.0.) IVBL=IX
      W=BS/AS(IVBL)
      Z=W*C(IVBL)
225   ITER=ITER + 1
      IF(ITER.GT.500) GO TO 998
C     WRITE(6,1000) Z,IVBL
1000  FORMAT(E12.5,'X(',I2,')')
C*****
C     CALCULATE NEW SURROGATE MULTIPLIERS
C
300   NOADJ=NOADJ+1
      CALL CHANGE
      IF(ICHNGE.GT.0) GO TO 997
      K=L
      GO TO 500
C*****

```

C OUTPUT RESULTS

```

C
998 WRITE(6,109)ITER
109 FORMAT(15X,'NO CONVERGENCE IN',I4,'ITERATIONS')
GO TO 999
997 WRITE(6,117)
WRITE(6,113) ITER,Z
WRITE(6,112) ((I,U(I,K)),I=1,M)
WRITE(6,117)
117 FORMAT(1X,70(1H*))
113 FORMAT(1X,'OPTIMALITY ESTABLISHED AT ITERATION',I4,
15X,'OBJ FUNCTION=',E12.5//10X,'SURROGATE MULTIPLIERS')
112 FORMAT(/,3(5X,'U(',I2,') = ',F10.7))
999 STOP
END

```

```

C
C
C
C

```

```

SUBROUTINE CHANGE
COMMON LHS(100),W,D(100),B(100),U(100,2),A(100,50)
COMMON K,L,ISET,Z,OBJ,AVG,IMIN,WMIN,IVBL,NOADJ,NODI,
1 Q0,D1,Q2,M,BS,TLRNCE,ICHNGE,IUNB,SBS,INEG
REAL LHS
IF(NOADJ.LT.15) GO TO 403
GO TO 399
404 DO 402 I=1,M
402 D(I)=D(I) + LHS(I) * 1.
K=3-L
ISET=0
AVG=1.E35
GO TO 303
403 IREP=0
IF(IUNB.EQ.0) GO TO 405
Q0=Q0/2.
IUNB=0
GO TO 311
405 IF(ISET.EQ.1) GO TO 400
IF(Z.LT.OBJ) GO TO 300
ISET=1
400 K=3-L
IF(Z.GT.AVG) GO TO 401
IF(INEG.GT.0)AVG=Z
IMIN=IVBL
WMIN=W
SBS=BS
Q0=Q0/2.
IF(Z.GT.OBJ.OR.ICHNGE.EQ.-1) GO TO 311
K=L
GO TO 300
401 ISET=0
ICHNGE=0

```

```

Z=AVG
IVBL=IMIN
W=WMIN
BS=SBS
Q0=Q0*2,
IREP=1
GO TO 311
410 IREP=0
K=L
300 IF(ABS(OBJ-Z).GE.TLRNCE) GO TO 301
IF(ISET.EQ.0) GO TO 399
IF(Z.EQ.OBJ) GO TO 399
ICHNGE=-1
K=3-L
GO TO 311
399 NODI=NODI+1
IF(NODI.GE.10) ICHNGE=1
IF(ICHNGE.LE.0) GO TO 404
RETURN
301 OBJ=Z
TLRNCE=ABS(.001*OBJ)
NODI=0
ISET=0
AVG=1.E35
DO 302 I=1,M
302 LHS(I) = A(I,IVBL)*W + 1. - B(I)
D(I) = D(I) + LHS(I) - 1.
L=3-K
303 NOADJ=0
Q0=0.
D1=0.
D2=0.
DO 309 I=1,M
IF(D(I)) 308,308,307
308 Q0=Q0 + U(I,K)/2.
D1=D1 + D(I)*U(I,K)
GO TO 309
307 D2=D2 + D(I)*U(I,K)
309 CONTINUE
311 DO 312 I=1,M
IF(D(I)) 320,330,310
320 U(I,L)=U(I,K) - D(I)*U(I,K)*Q0/D1
GO TO 312
330 U(I,L)=U(I,K)
GO TO 312
310 U(I,L)=U(I,K) + D(I)*U(I,K)*Q0/D2
312 CONTINUE
INEG=1
DO 314 I=1,M
IF(U(I,L)) 323,323,314
323 U(I,L)=U(I,K)/4.

```

```
      INEG=-1
314  CONTINUE
      IF(INEG.GT.0) GO TO 316
      QSUM=0.
      DO 317 I=1,M
      IF(D(I)) 318,317,317
318  QSUM=QSUM + U(I,K) - U(I,L)
317  CONTINUE
      DO 319 I=1,M
      IF(D(I))319,319,329
329  U(I,L)=U(I,K) + D(I)*U(I,K)*QSUM/D2
319  CONTINUE
316  IF(IREP.EQ.1) GO TO 410
      RETURN
      END
```

```

C*****
C  SIMPLEX METHOD (GIVEN A BASIC FEASIBLE SOLUTION)
C
C
C*****
      DIMENSION A(50,50),IB(50)
      DOUBLE PRECISION FMTA(3),FMTB(3)
      READ(5,100) N,M
100   FORMAT(2G)
      READ(5,101) (FMTA(I),I=1,3)
      READ(5,101) (FMTB(I),I=1,3)
101   FORMAT(3A10)
      DO 1 I=1,M
1     READ(5,FMTA) (A(I,J),J=1,N)
      READ(5,FMTB) (IB(I),I=1,M)
      NPIVOT=0
11    AMIN=1.E35
      DO 2 J=1,N
        IF(A(1,J).GE.AMIN) GO TO 2
        AMIN=A(1,J)
        IPC=J
2     CONTINUE
        IF(AMIN.GE.0.) GO TO 99
        AMIN=1.E35
        DO 3 I=2,M
          IF(A(I,IPC).LE.0.) GO TO 3
          IF(A(I,N)/A(I,IPC).GE.AMIN) GO TO 3
          AMIN=A(I,N)/A(I,IPC)
3     CONTINUE
        IB(IPR)=IPC-1
        DO 10 I=1,M
          IF(I.EQ.IPR) GO TO 10
          FMULT=-A(I,IPC)/A(IPR,IPC)
          DO 9 J=1,N
8          A(I,J)=A(IPR,J)*FMULT + A(I,J)
10    CONTINUE
        FMULT=1./A(IPR,IPC)
        DO 12 J=1,N
12    A(IPR,J)=A(IPR,J)*FMULT
        NPIVOT=NPIVOT + 1
        GO TO 11
99    WRITE(6,102) NPIVOT
102   FORMAT(1X,15(1H*),' OPTIMALITY ESTABLISHED AFTER',13,' PIVOTS',
115(1H*))
      WRITE(6,103) A(1,N)
103   FORMAT(1X,'OBJECTIVE FUNCTION =',F20,5)
      WRITE(6,104)
      WRITE(6,105) (IB(I),A(I,N),I=1,M)
104   FORMAT(1X,'BASIC VARIABLES AT OPTIMALITY:')
105   FORMAT(5X,'X(',12,') =',F15,5)

```

ARTHUR LAKES LIBRARY
 COLORADO SCHOOL of MINES
 GOLDEN, COLORADO 80401

```

C*****
C  PROPOSED METHOD OF CHANGING THE SURROGATE MULTIPLIERS
C
C
C*****
      DIMENSION C(50),AS(50),LHS(100),B(100),U(100,2),A(100,50)
      DIMENSION UPOBJ(50)
      DOUBLE PRECISION FMTC(3),FMTE(3)
      REAL LHS
100  FORMAT(2G)
101  FORMAT(3A10)
102  FORMAT(55H
      READ(5,102)
      WRITE(6,102)
      READ(5,100) N,M
      READ(5,100) INPUT
      READ(5,101) (FMTC(I),I=1,3)
      READ(5,101) (FMTE(I),I=1,3)
      IF(INPUT.NE.0) READ(5,FMTE) (U(I,1),I=1,M)
      READ(5,FMTC) (C(J),J=1,N)
      READ(5,FMTE) (B(I),I=1,M)
      DO 2 I=1,M
2    READ(5,FMTC) (A(I,J),J=1,N)
C*****
C  FORM NORMALIZED PROBLEM AND INITIALIZE VBLS
C
      DO 3 I=1,M
      IF(B(I).EQ.0.) GO TO 3
      DO 4 J=1,N
      UPOBJ(J)=C(J)
4    A(I,J)=A(I,J)/ABS(B(I))
      B(I)=B(I)/ABS(B(I))
3    CONTINUE
      ISET=0
      AVG=1.E35
      OBJ=1.E35
      ITER=0
      K=1
C*****
C  INITIALIZE SURROGATE MULTIPLIERS
C
      IF(INPUT.NE.0) GO TO 500
      DO 5 I=1,M
5    U(I,K)=1.
C*****
C  FORM SURROGATE CONSTRAINT
C
500  IN=0
      IX=0
      BS=0.
      DO 50 J=1,N

```

```

50   AS(J)=0,
      DO 6 J=1,N
      DO 6 I=1,M
6     AS(J)=AS(J) + U(I,K)*A(I,J)
      DO 7 I=1,M
7     BS=BS + B(I)*U(I,K)
C*****
C   SOLVE THE SURROGATED PROBLEM
C
      IPA=0
      INA=0
      IPC=0
      INC=0
      AMAX=0.
      AMIN=1.E35
      DO 200 J=1,N
      IF(C(J)) 221,222,222
221  INC=-1
      IF(AS(J).GE.0.) GO TO 220
      INA=-1
      IF(C(J)/AS(J).GE.AMIN) GO TO 200
      AMIN=C(J)/AS(J)
      IN=J
      GO TO 200
220  IPA=1
      GO TO 200
222  IPC=1
      IF(AS(J).LE.0.) GO TO 223
      IPA=1
      IF(C(J)/AS(J).LE.AMAX) GO TO 200
      AMAX=C(J)/AS(J)
      IX=J
200  CONTINUE
      IA=IPA + INA
      IC=IPC + INC
      IF(IC.LT.0.AND.BS.GT.0.) GO TO 14
      IF(IC.LT.0.AND.IA.LT.1.AND.BS.EQ.0.) GO TO 14
      IF(IA.GT.0.AND.BS.EQ.0.) GO TO 14
      IF(IA.GT.0.AND.BS.LT.0.) GO TO 15
      IF(AMIN.GE.AMAX) GO TO 224
223  IF(ITER.NE.0) GO TO 501
      WRITE(6,130)
130  FORMAT(15X,'UNBOUNDED INITIAL SURROGATE VECTOR')
      STOP
14   Z=0.
      IVBL=1
      W=0.
      GO TO 225
15   WRITE(6,105)
105  FORMAT(15X,'SURROGATE PROBLEM HAS NO FEASIBLE SOLN')
      STOP

```

```

224  IVBL=IN
      IF(BS.GE.0.) IVBL=IX
      W=BS/AS(IVBL)
      Z=W*C(IVBL)
225  ITER=ITER + 1
      IF(ITER.GT.200) GO TO 998
C     WRITE(6,1000) Z,IVBL
1000  FORMAT(E12.5,'X(',I2,')')
C*****
C     CALCULATE NEW SURROGATE MULTIPLIERS
C
      IF(ISET.GT.0) GO TO 501
      IF(Z.LT.OBJ) GO TO 550
501   ISET=ISET + 1
      K=3-L
      IF(Z.GT.AVG.OR.ISET.GT.50) GO TO 502
      AVG=Z
      IVAR=IVBL
      RK2=RK1 + 2.
      GO TO 591
502   Z=AVG
      RK2=RK1 - 2.
      DO 503 I=1,M
503   LHS(I)=(LHS(I)-1.)*RK1/RK2 + 1.
      DO 504 I=1,M
504   U(I,L)=U(I,K)*LHS(I)
      K=L
      GO TO 900
550   IF(ABS(OBJ-Z).LT.TLRNCE) GO TO 997
      OBJ=Z
      RATIO=UPOBJ(IVBL)/AS(IVBL)
      DO 551 J=1,N
551   UPOBJ(J)=UPOBJ(J) - AS(J)*RATIO
      TLRNCE=ABS(.001*OBJ)
      RK1=1.
      DO 562 I=1,M
562   LHS(I)=A(I,IVBL)*W + 1. - B(I)
      L=3-K
      DO 563 I=1,M
563   IF(LHS(I).LE.0.) GO TO 570
564   DO 565 I=1,M
565   U(I,L)=U(I,K)*LHS(I)
      K=L
      GO TO 500
570   AMIN=0.
      DO 577 I=1,M
577   IF(LHS(I).LT.AMIN) AMIN=LHS(I)
      RK2=IFIX(-AMIN) + 2.
591   DO 598 I=1,M
598   LHS(I)=(LHS(I)-1.)*RK1/RK2 + 1.
      RK1=RK2

```

```

      GO TO 564
C*****
C  OUTPUT RESULTS
C
998  WRITE(6,109)ITER
      GO TO 999
900  DO 901 J=1,N
901  AS(J)=0.
      DO 902 J=1,N
      DO 902 I=1,M
902  AS(J)=AS(J) + U(I,K)*A(I,J)
      IVBL=IVAR
997  WRITE(6,117)
      WRITE(6,113) ITER,Z
      WRITE(6,112) ((I,U(I,K)),I=1,M)
      RATIO=UPOBJ(IVBL)/AS(IVBL)
      DO 950 J=1,N
950  UPOBJ(J)=ABS(UPOBJ(J) - AS(J)*RATIO)
      DO 960 J=1,N
      AMIN=1.E35
      DO 955 JJ=1,N
      IF(UPOBJ(JJ).GE.AMIN) GO TO 955
      AMIN=UPOBJ(JJ)
      ID=JJ
955  CONTINUE
      LHS(J)=ID
960  UPOBJ(ID)=1.E35
      IX=0
      B(1)=0.
      L=3-K
      U(1,L)=0.
      IY=0
      DO 995 I=1,M
      IF(U(I,K).LE.1.5) GO TO 994
      IX=IX+1
      B(IX)=I
994  IF(U(I,K).GE..5) GO TO 995
      IY=IY+1
      U(IY,L)=I
995  CONTINUE
      INDX=MIN0(IX,N)
      WRITE(6,114)
      WRITE(6,118) (U(I,L),I=1,IY)
      WRITE(6,115)
      WRITE(6,118) (B(I),I=1,IX)
      WRITE(6,116)
      WRITE(6,118) (LHS(J),J=1,INDX)
      WRITE(6,117)
109  FORMAT(15X,'NO CONVERGENCE IN',I4,'ITERATIONS')
117  FORMAT(1X,70(1H*))
113  FORMAT(1X,'OPTIMALITY ESTABLISHED AT ITERATION',I4,

```

```
15X,'OBJ FUNCTION=',E12.5//10X,'SURROGATE MULTIPLIERS')
112  FORMAT(/,3(5X,'U(',I2,') = ',F10.7))
114  FORMAT(// ' LOOSE CONSTRAINTS:')
115  FORMAT(// ' TIGHT CONSTRAINTS:')
116  FORMAT(// ' ORDERED CANDIDATES FOR BASIC VARIABLES:')
118  FORMAT(20F3.0,/)
999  STOP
      END
```

DITTMAN EXAMPLE PROBLEM 1

2,9

0

(2G)

(9G)

1,-2

1,1,1,1,1,1,1,1,1

3,4

5,6

3,7

8,9

4,3

9,8

9,1

1,1

2,10

DITTMAN EXAMPLE PROBLEM 2

3,14

0

(3G)

(14G)

1,-3,1

1,1,1,1,1,1,1,1,1,1,1,1,1,1

1,2,1

2,3,1

4,1,6

3,1,4

5,1,3

3,6,1

4,1,3

16,14,15

10,12,16

36,14,8

17,4,3

5,1,6

4,6,8

1,2,3

DITTMAN EXAMPLE PROBLEM 5

7,4

∅

(7G)

(4G)

1,6,7,1,1,1,1

1,1,1,-1

4,3,1,5,1,1,1

3,-2,3,5,1,6,1

2,3,1,1,3,1,6

-2,-3,3,5,-1,6,6

DITTMAN EXAMPLE PROBLEM 6

5,4

1

(5G)

(4G)

1.75,,25,,25,1.75

5,6,7,8,1

1,1,1,1

6,6,1,1,6

1,4,-7,1,1

5,3,-4,-3,2

3,1,1,2,3

DITTMAN EXAMPLE PROBLEM 11

2,2
0
(2G)
(2G)
7,6
1,1
1,6
5,1

DITTMAN EXAMPLE PROBLEM 13

4,4

0

(4G)

(4G)

1,2,-3,-1

1,1,1,1

1,2,1,1

4,3,1,2

5,1,3,1

1,2,3,1

DITTMAN EXAMPLE PROBLEM 15

7,3

Ø

(7G)

(3G)

1,2,3,4,5,6,7

1,1,1

3,2,4,1,5,1,6

4,3,1,5,6,6,1

5,5,1,4,1,1,7

DITTMAN EXAMPLE PROBLEM 19

7,7

0

(7G)

(7G)

4,1,4,1,4,1,4

1,1,1,1,1,1,1

3,1,2,1,1,6,1

2,2,1,3,1,1,7

5,1,3,4,1,6,7

1,1,1,1,1,1,1

2,2,2,2,2,2,2

3,3,3,3,3,3,3

4,4,4,4,4,4,4

DITTMAN EXAMPLE PROBLEM 20

3,7

0

(3G)

(7G)

1,1,1

1,1,1,1,1,1,1

1,4,3

5,2,3

6,1,5

4,3,1

1,6,4

7,5,1

2,3,2

ARTHUR LAKES LIBRARY
COLORADO SCHOOL of MINES
GOLDEN, COLORADO 80401

SYMONDS FUEL OIL BLENDING PROBLEM-PAGE 4 (EX.1)
3,3
0
(30)
(30)
5,4,-3
0,0,1000
.0002,.0001,-.0002
.0001,.0001,-.0003
1,1,0

SYMONDS RUNNING PLAN PROBLEM-PAGE 12 (EX.2)

5,8

0

(5G)

(8G)

100,200,150,250,70

100,100,200,100,170,85,20,85

1,0,0,0,0

0,1,0,0,0

0,0,1,1,0

0,0,0,0,1

.6,.5,.4,.4,.3

.2,.2,.3,.1,.3

0,0,0,.2,0

.1,.2,.2,.2,.3

SYMONDS BLENDING AVIATION GASOLINE-PAGE 18 (EX.3)

8,10

0

(8G)

(10G)

0,2,1,3,2,4,3,5

2000,2000,1000,1000,1000,1000,0,0,2,0

1,0,1,0,1,0,1,0

0,1,0,1,0,1,0,1

1,1,0,0,0,0,0,0

0,0,1,1,0,0,0,0

0,0,0,0,1,1,0,0

0,0,0,0,0,0,1,1

-.0002,0,-.0001,0,0,0,.0001,0

-.0005,0,-.0003,0,.0003,0,.0001,0

0,-.0001,0,.0001,0,.0003,0,.0005

0,-.0001,0,0,0,.0003,0,.0002

HILLIER=LIEBERMAN MANUFACTURING PROBLEM-PAGE 129 (EX.1)

3,4

0

(3G)

(4G)

20,6,8

200,100,50,20

8,2,3

4,3,0

2,0,1

0,0,1

HILLIER-LIEBERMAN PRODUCT MIX PROBLEM-PAGE 131 (EX.2)

12,10

0

(12G)

(10G)

2,5,-.5,1,5,.5,1,5,-1,5,.5,-.5,.5,-2,5,-.5,-1,5

3000,2000,4000,1000,0,0,0,0,0,0

1,0,0,0,1,0,0,0,1,0,0,0

0,1,0,0,0,1,0,0,0,1,0,0

0,0,1,0,0,0,1,0,0,0,1,0

0,0,0,1,0,0,0,1,0,0,0,1

.00007,-.00003,-.00003,-.00003,0,0,0,0,0,0,0,0

.00004,-.00006,.00004,.00004,0,0,0,0,0,0,0,0

-.00005,-.00005,.00005,-.00005,0,0,0,0,0,0,0,0

0,0,0,0,.00005,-.00005,-.00005,-.00005,0,0,0,0

0,0,0,0,.00001,-.00009,.00001,.00001,0,0,0,0

0,0,0,0,0,0,0,0,.00003,-.00007,-.00007,-.00007

HILLIER-LIEBERMAN FARM OPERATION-PAGE 134 (EX.3)

9,13

0

(9G)

(13G)

400,400,400,300,300,300,100,100,100
400,600,300,1500,2000,900,700,800,300,0,0,0,0
1,0,0,1,0,0,1,0,0
0,1,0,0,1,0,0,1,0
0,0,1,0,0,1,0,0,1
5,0,0,4,0,0,3,0,0
0,5,0,0,4,0,0,3,0
0,0,5,0,0,4,0,0,3
1,1,1,0,0,0,0,0,0
0,0,0,1,1,1,0,0,0
0,0,0,0,0,0,1,1,1
.003,-.002,0,.003,-.002,0,.003,-.002,0
-.003,.002,0,-.003,.002,0,-.003,.002,0
0,.001,-.002,0,.001,-.002,0,.001,-.002
0,-.001,.002,0,-.001,.002,0,-.001,.002

DANØ INVENTORY PROBLEM-PAGE 57

8,13

1

(8G)

(13G)

.5,.5,.5,.5,1.25,1.25,1.25,1.25,1.25,1.25,1.25,1.25,1.25,1.0

63,65,62.64,61.63,60.62,76.69,75.68,74.67,73.66

-712,-1290,-1837,-3053,3053,550.5,550.5,550.5,550.5,453,453,453,453

-1,0,0,0,-1,0,0,0

-1,-1,0,0,-1,-1,0,0

-1,-1,-1,0,-1,-1,-1,0

-1,-1,-1,-1,-1,-1,-1,-1

1,1,1,1,1,1,1,1

1,0,0,0,0,0,0,0

0,1,0,0,0,0,0,0

0,0,1,0,0,0,0,0

0,0,0,1,0,0,0,0

0,0,0,0,1,0,0,0

0,0,0,0,0,1,0,0

0,0,0,0,0,0,1,0

0,0,0,0,0,0,0,1

CHARNES&COOPER MACHINE LOADING-PAGE 25 (EX.1)

21.15

0

(11G/10G)

(10G/5G)

80.76,70.77,76.15,65.3,79.11,119.49,89.09,75.3,116.3,91.49,102.11

64.14,54.13,62.98,64.14,86.25,45.57,29.03,39.87,42.05,38.23

6864,6864,6864,6864,6864,126151,11714,11463,10776,5679

3410,2834,1109,948,144

,534,0,0,0,0,0,0,0,0,0,0

0,0,.448,0,0,.427,0,0,.496,.440

0,.511,0,0,0,.480,0,0,0,.473,0

0,0,0,0,0,0,0,0,0,0,0

0,0,.471,0,0,0,.415,0,0,0,.424

,376,0,0,0,.436,0,0,0,0,0,0

0,0,0,.501,0,0,0,.469,0,0,0

0,0,0,0,0,0,0,.456,0,0,0

0,0,0,0,.246,0,0,0,.245,0,0

0,.183,0,.236,0,0,0,.248,0,0

1,1,1,1,1,0,0,0,0,0,0

0,0,0,0,0,0,0,0,0,0,0

0,0,0,0,0,1,1,1,1,0,0

0,0,0,0,0,0,0,0,0,0,0

0,0,0,0,0,0,0,0,0,1,1

0,0,0,0,0,0,0,0,0,0,0

0,0,0,0,0,0,0,0,0,0,0

1,1,0,0,0,0,0,0,0,0,0

0,0,0,0,0,0,0,0,0,0,0

0,0,1,1,0,0,0,0,0,0,0

0,0,0,0,0,0,0,0,0,0,0

0,0,0,0,1,0,0,0,0,0,0

0,0,0,0,0,0,0,0,0,0,0

0,0,0,0,0,1,1,0,0,0,0

0,0,0,0,0,0,0,0,0,0,0

0,0,0,0,0,0,0,1,0,0,0

0,0,0,0,0,0,0,0,0,0,0

0,0,0,0,0,0,0,0,1,0,0

0,0,0,0,0,0,0,0,0,0,0

0,0,0,0,0,0,0,0,0,1,0

0,0,0,0,0,0,0,0,0,0,1

CHARNES&COOPER BLENDING MODEL-PAGE 552 (EX.2)

8,8

0

(8G)

(8G)

.9,.9,.9,.9,1.5,1.5,1.5,1.5

0,0,0,0,3800,2652,4081,1300

-.00016,-.00002,.00004,-.00017,0,0,0,0

-.00002,.00001,-.00003,.00013,0,0,0,0

0,0,0,0,-.00007,.00007,.00013,-.00008

0,0,0,0,-.00002,.00001,-.00003,.00013

1,0,0,0,1,0,0,0

0,1,0,0,0,1,0,0

0,0,1,0,0,0,1,0

0,0,0,1,0,0,0,1

CHARNES&COOPER FOUR-FIELD PIPELINER-PAGE 588 (EX.3)

16,24

0

(16G)

(12G/12G)

.80,.76,.72,.65,.76,.72,.68,.60,.72,.67,.60,.50,.60,.58,.56,.54
900,900,900,900,1200,1200,1200,1200,1300,1300,1300,1300
1800,1800,1800,1800,27700,11100,33000,483300,20000,20000,20000,20000
.01677,0,0,0,0,0,0,0,0,0,0,0,0,0,0
.00057,.01677,0,0,0,0,0,0,0,0,0,0,0,0,0
.00033,.00057,.01677,0,0,0,0,0,0,0,0,0,0,0,0
.00023,.00033,.00057,.01677,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,.04687,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,.00238,.04687,0,0,0,0,0,0,0,0,0,0
0,0,0,0,.00096,.00238,.04687,0,0,0,0,0,0,0,0,0
0,0,0,0,.00091,.00096,.00238,.04687,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,.16477,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,.00626,.16477,0,0,0,0,0,0
0,0,0,0,0,0,0,0,.00358,.00626,.16477,0,0,0,0,0
0,0,0,0,0,0,0,0,.00277,.00358,.00626,.16477,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,.001513,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,.000478,.001513,0,0
0,0,0,0,0,0,0,0,0,0,0,0,.000277,.000478,.001513,0
0,0,0,0,0,0,0,0,0,0,0,0,.000161,.000277,.000478,.001513
1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1
1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0
0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0
0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0
0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1