NPS55-82-009

# NAVAL POSTGRADUATE SCHOOL

## Monterey, California

# A HEURISTIC FOR CONSTRUCTING SURROGATE CONSTRAINTS FOR THE LINEAR ZERO-ONE INTEGER PROGRAMMING PROBLEM

by

Frank R. Giordano

February 1982



Prepared for:
Naval Postgraduate School
Monterey, ca 93940

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

Rear Admiral J. J. Ekelund
Superintendent

David A. Schrady
Acting Provost

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

# REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

| | |
|---|---|
| 1. REPORT NUMBER: NPS55-82-009 | 2. GOVT ACCESSION NO. |
| 3. RECIPIENT'S CATALOG NUMBER | |
| 4. TITLE (and Subtitle): A HEURISTIC FOR CONSTRUCTING SURROGATE CONSTRAINTS FOR THE LINEAR ZERO-ONE INTEGER PROGRAMMING PROBLEM | 5. TYPE OF REPORT & PERIOD COVERED: Technical |
| | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s): Frank R. Giordano | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, CA 93940 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS | 12. REPORT DATE: February 1982 |
| | 13. NUMBER OF PAGES: 19 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) | 15. SECURITY CLASS. (of this report): UNCLASSIFIED |
| | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this report the author presents a heuristic for constructing surrogate constraints to be used for the solution of the linear zero-one integer problem. Using the heuristic the author was able to build surrogate constraints with strength comparable to the dual multiplier surrogate in one-tenth the time.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

# 1. BACKGROUND

## 1.1 DEFINITIONS

Given the binary linear integer programming problem

$$\text{Minimize } \{cx \mid Ax \leq b,\ x = (0,1)\} \tag{1}$$

where A is an m by n matrix, a surrogate constraint for problem (1) is defined as follows:

$$u(b-Ax) \geq 0, \quad u = (u_1, u_2, \ldots, u_m), \quad u \geq 0. \tag{2}$$

We will also find it convenient to define a corresponding one-constraint (knapsack) surrogate problem:

$$\text{Minimize } \{cx \mid u(b-Ax) \geq 0,\ u \geq 0,\ x = (0,1)\}. \tag{3}$$

## 1.2 PROPERTIES

Because the vector $u \geq 0$ implies that the solution set to inequality (2) contains the solution set to (1), we have the following properties:

1. If x* is a feasible solution to (1) it is also feasible to (2) and (3).
2. If the surrogate constraint (2) has no feasible solution, then neither does the original problem (1).

## 1.3 USES

Surrogate constraints are used in conjunction with implicit enumeration algorithms (e.g., Balas) in several ways. Each vertex in an enumeration tree represents a restriction of problem (1). Problems (1), (2) and (3) can be written in explicit terms of the restriction being studied by substitution of the variables assigned values by the restriction. If it can be shown that a surrogate constraint corresponding to a vertex has no feasible solution (completion), then the vertex being studied can be fathomed by Property (2). If the vertex cannot be fathomed, Property (1) allows the surrogate to be appended to the constraint set as a valid constraint to be used with the implicit enumeration tests to identify promising variables for further exploration. Further, if the solution to Problem (3) corresponding to the vertex restriction is known, it provides an upper bound on the original problem (1) if it is feasible for (1) or a lower bound on the vertex restriction if it is infeasible for (1).

## 1.4 STRENGTH OF SURROGATES

In order to choose $u \geq 0$, Glover [1965] defined surrogate $u^1(b-Ax) \geq 0$ to be stronger than $u^2(b-Ax) \geq 0$ if

$$\text{Min } \{cx \mid u^1(b-Ax) \geq 0,\ x = (0,1)\} > \text{Min } \{cx \mid u^2(b-Ax) \geq 0,\ x = (0,1)\}$$

$$\text{for } u^1,\ u^2 \geq 0. \qquad (4)$$

This definition states that if the corresponding surrogate

knapsack problems (3) are resolved, the surrogate resulting in the more restrictive lower bound to problem (1) is stronger. Essentially, that surrogate eliminating more solutions (as measured by the objective function) is the stronger. This is intuitively appealing since by Property (1) the surrogate cannot eliminate any feasible solutions to the original problem. Thus by this definition we should choose u as follows:

$$\underset{u\geq 0}{\text{Max}} \ \underset{x=(0,1)}{\text{Min}} \ \{cx \mid u(b-Ax) \geq 0\} \tag{5}$$

Unfortunately (5) is difficult to solve for general cases, although Glover [1965] has studied the two constraint case. An approximation to (5) can be made by relaxing the integer restriction on x, i.e., choose $u \geq 0$ satisfying

$$\underset{u\geq 0}{\text{Max}} \ \underset{0\leq x<1}{\text{Min}} \ \{cx \mid u(b-Ax) \geq 0\} \tag{6}$$

In a previous report the strength of the approximation (6) as measured by (4) was studied, Giordano [1982]. In this report we will present an alternative approximation to (5) and compare both the strength and speed of the alternative approximation to the approximation suggested by (6).

## 2. THE DUAL MULTIPLIER SURROGATE

### 2.1 DEFINITION

The advantage of the relaxation to (6) is that it can be resolved optimally yielding $u^o = v^o$ where $v^o$ are the dual

variables to the linear programming (LP) relaxation of (1). Thus at any given vertex restriction, after substituting the variables assigned values, the LP written in terms of the remaining free variables may be resolved and the optimal values of the dual variables used as weights to form a surrogate constraint. A surrogate so formed is called a dual multiplier surrogate.

## 2.2 PROPERTIES/USES

As with all surrogates, if it can be shown that the dual multiplier surrogate has no feasible solution then the vertex can be fathomed. This test can be strengthened by requiring that the solution to the surrogate constraint also improve the current upper bound on problem (1), Geoffrion [1969]. After resolving the corresponding LP for the dual variables, the value of the free variables may be used to solve the LP relaxation of (3) directly. Note in this case when solving for the dual variables we are solving the LP relaxation of (1) corresponding to the vertex restriction. If the values of the free variables are integer, problem (1) has been solved for that vertex and a new upper bound on the original problem has been obtained. If the values are fractional, then a lower bound for the vertex is obtained. If desired, heuristics may be applied to the fractional values to identify promising variables for branching.

## 2.3 COMPUTATIONAL ADVANTAGES

The dual multiplier surrogate has been widely used in conjunction with implicit enumeration algorithms and research has been conducted on frequency of use, maximum number of constraints to carry forward, and related matters. It is interesting to note the effect of the use of the dual multiplier surrogate on the problem set studied, which includes 18 problems with up to 50 variables and up to 10 constraints. Seventeen of the problems required a total of 552.86 CPU seconds (CDC 6500) using a Balas Algorithm with heuristics. The same Balas Algorithm employing a dual multiplier surrogate generated every eight iterations and carrying a maximum of four surrogates forward solved the 17 problems in 32.15 CPU seconds. Problem 18, consisting of 50 variables and 5 constraints, had not been solved optimally after 5631 CPU seconds using the Balas Algorithm but was solved optimally in 6.47 seconds when the surrogate was added. The results are summarized in Table 1, which is found at the end of this report.

## 2.4 OBSERVATIONS

The dual multiplier surrogate has been a very significant contribution to implicit enumeration. Nevertheless, there are disadvantages inherent in the dual multiplier surrogate when applied to large problems. A linear program must be solved at each vertex at which a surrogate constraint is to be formed. As problems with larger numbers of variables are considered, not only does the size of the corresponding LP's increase, but more importantly, the number of vertices grows exponentially.

Since one of the primary advantages of the Balas Algorithm is that it is additive computationally, the necessity of solving the LP's should be investigated. Note that the necessity to solve the LP's makes the integer programming problem more sensitive to the number of constraints than is otherwise the case. Ideally one would like to build a surrogate with strength and computational advantages comparable to the dual multiplier surrogate but requiring less computation time.

## 3. AN ALTERNATIVE METHOD FOR CONSTRUCTING SURROGATE CONSTRAINTS

Definition (4) suggests an alternative strategy for constructing surrogates. Given an initial surrogate a stronger surrogate can be constructed by making the optimal solution to the current surrogate knapsack problem infeasible for the new surrogate constraint while continuing to eliminate less optimal solutions. The process iterates until a stronger surrogate can no longer be constructed. Such an iterative procedure was developed when the strength of surrogates was investigated, Giordano [1982]. In the referenced report, the initial surrogate was the dual multiplier surrogate and each surrogate knapsack problem was resolved for an optimal solution using a Balas Algorithm.

To develop a quick heuristic for constructing surrogates, two major problems must be resolved. First an alternative method for forming the initial surrogate must be developed since solving the corresponding LP at each vertex is

computationally costly. Secondly, an alternative process for solving the surrogate knapsack problems must be incorporated to avoid the computation time involved in the Balas Algorithm. We will review the procedure for iterating to a best surrogate, investigate alternative methods for forming the initial surrogate, present an approximation technique for resolving the surrogate knapsack problems, and compare the formation time and strength of the heuristically generated surrogate with the dual multiplier surrogate.

## 4. ITERATING TO A BEST SURROGATE

### 4.1 AN ALGORITHM

Let us define for the current surrogate:

$x^*$: the optimal solution to the current surrogate knapsack problem.

$u_i^*$: the weight of the ith constraint in the current surrogate.

$s_i^*$: the slack $x^*$ allows in the ith constraint.

$S_0^*$: $\sum_i u_i^* s_i^*$: the slack $x^*$ allows in the current surrogate.

Similarly, for the new surrogate let:

$u_i'$: the weight of the ith constraint in the new surrogate.

$S_0'$: $\sum_i u_i' s_i^*$: the slack $x^*$ would allow in the new surrogate.

Let:

$$u_i' = u_i^* - \alpha_i \theta .$$

The purpose of $\alpha_i$ is to increase the weight of the constraints violated by $x^*$ and $\theta$ is a parameter to insure $x^*$ becomes infeasible for the new surrogate. Choosing

$$\alpha_i = s_i^*$$

and $$\theta = (S_0^* / \sum_i s_i^{*2}) + .05$$

a new surrogate is generated and combined with the previous surrogate, weighting the previous surrogate 75%. If a successive surrogate fails to be stronger than its predecessor, $\varepsilon = .05$ above is halved and the process repeated. If a surrogate fails to improve after three reductions, the process terminates with the previous surrogate judged 'best'.

## 4.2 COMPARING THE STRENGTH OF SURROGATES

The strength of a surrogate is measured by the optimal solution to the corresponding knapsack problem. To compare various surrogates the following measure proved convenient:

$$\text{percent convergence} = |z_0' - z_0''| / |z_0' - z_0|$$

where:

$z_0$: objective function value of the optimal solution to problem (1).

$z_0'$: objective function value of the optimal solution to the continuous relaxation to (1).

$z_0''$: objective function value of the optimal solution to (3) for the surrogate in question.

The percent convergence heuristically measures the number of infeasible solutions between the continuous and integer optimum solutions to (1) that a particular surrogate effectively eliminates and is an indication of the relative strength of two surrogates.

### 4.3 RESULTS

Of the 18 problems considered, the solution to the dual multiplier surrogate knapsack problem solves the original problem directly in 7 cases. In the remaining 11 cases, it is possible to build a stronger surrogate in 9 cases. The improvement in most instances is substantial. In fact, the best surrogate obtained solves an additional 4 problems. The results are summarized in Table 2, which is located at the end of the report.

## 5. THE INITIAL SURROGATE

### 5.1 METHODS TESTED

Three alternative methods for forming the initial surrogate were tested:

1. Simply adding the original constraints.
2. Averaging the original constraints.
3. Weighting each of the original constraints according to the right hand side.

## 5.2 DISCUSSION

The advantage of Method 1 is that it is a simple way of getting started. Method 2 attempts to prevent the initial surrogate from becoming so large that it compromises the weighting scheme developed in 4.1 when forming subsequent surrogates. Method 3 attempts to exploit the format of the problem for the vertex being studied. In the Balas format, the problem is better than optimal with all variables at the zero level. Variables are raised to the one level only to cure infeasibility. For violated constraints, the current right hand side represents the infeasibility which must be "cured". Using Method 3, the initial surrogate is formed weighting each violated constraint according to its relative infeasibility. Various normalization schemes were tested to insure a unit of slack in each constraint means approximately the same thing.

## 5.3 RESULTS

For the problem set tested, Method 2 generally produces the best results. Although the initial surrogate is normally not as strong as the dual multiplier surrogate, the process quickly converges to a "best" surrogate. Since the initial surrogate is not as strong as the dual multiplier surrogate, some loss of strength is experienced. The best surrogate using the dual multiplier surrogate as the initial surrogate is equaled in 12 of the 18 problems tested. More importantly, the best surrogate generated using Method 2 above equals or

improves the strength of the dual multiplier surrogate in 15 of the 18 cases. The results are summarized in Table 2.

## 6. SOLVING THE SURROGATE KNAPSACK PROBLEMS

### 6.1 AN APPROXIMATE ALGORITHM

Consider the knapsack problem:

$$\text{Minimize } \{\sum_j (-c_j x_j) \mid \sum_j a_j x_j \leq b,\ x_j = 0 \text{ or } 1\}.$$

By preassigning values for $x_j$ where $c_j$ and $a_j$ differ in sign and substituting $x_j = 1 - x_j'$ for the remaining variables where $c_j$ and $a_j$ are both negative, the above problem can be reduced to a form with all $c_j$, $a_j$, and b positive. Arrange the indices of $x_j$ such that $c_1/a_1 \geq c_2/a_2 \geq \ldots \geq c_n/a_n$. We will refer to $c_j/a_j$ as the knapsack ratio for $x_j$. Find p the least integer $(0 \leq p \leq n)$ such that $\sum_{j \leq p} a_j \geq b$. Beginning with $r = p$ increment r by unit steps to n, adding each $a_r$ to $\sum_{k=1}^{p-1} a_k$ if and only if the resultant cumulative sum is less than or equal to b. Then the approximate solution is given by

$$x_j^* = \begin{cases} 1, & \text{if } j < p \text{ or if } a_j \text{ is added to the summation.} \\ 0, & \text{otherwise.} \end{cases}$$

This algorithm arranges the variables in such a manner that the more attractive variables are elevated to level one before infeasibility occurs, Taha [1975].

## 6.2 INCORPORATING THE APPROXIMATE KNAPSACK ALGORITHM

Since, in the algorithm presented in 4.1, it is only necessary that a subsequent surrogate be relatively stronger than a previous surrogate, approximate measurements of their strengths should be sufficient. When the approximate algorithm is incorporated, the desired speed is obtained, some loss in strength in the 'best' surrogate is experienced, fewer iterations are required to converge, and the probability of finding a feasible solution to the original problem increases. Part of the loss in strength of the best surrogate is due to terminating the process when a feasible solution to the original problem is found. For example, in Problem 4, the approximate solution to the best surrogate constructed solves the original problem.

Since the approximate solution to a surrogate constraint is greater (minimization) than the exact solution, such a solution poses a greater restriction on the subsequent surrogate. This reduces the number of iterations required to obtain a best surrogate and reduces the probability of building stronger surrogates in the vicinity of the best surrogate, since no surrogate can be constructed 'between' the current surrogate and the approximate solution to the current surrogate. The fact that the approximate solution is greater than the exact solution to the surrogate also explains the increase in the number of feasible solutions to the original problem found. The computational advantage of using these feasible solutions in the Balas Algorithm will be subsequently discussed.

### 6.3 RESULTS

In the 18 problems tested, the best surrogate obtained equals the dual multiplier surrogate in 10 cases, is stronger in 4 cases, and weaker in 4 cases. Thus the two surrogates are roughly equivalent in strength. However, the heuristically determined best surrogate is formed in a total of .89 seconds for the problem set compared with 9.19 seconds for the dual multiplier surrogate. The results are summarized in Table 2.

## 7. ANALYSIS OF ANCILLARY INFORMATION

### 7.1 TERMINATION OF ALGORITHM

The algorithm presented in 4.1, modified to incorporate Method 2 for generating an initial surrogate and the approximate technique for resolving the surrogate knapsack problems, terminates under the following conditions:

1. Stronger surrogates can no longer be constructed.
2. A feasible solution to the original problem has been found.

Upon termination, one always has a solution which is a lower bound for the vertex. The best surrogate formed is a weighted combination of the original constraints which more heavily weights those constraints which, in some sense, are critical. The final set of knapsack ratios thus represents the attractiveness of the variables with respect to the critical constraints. One can take advantage of this situation to attempt to find feasible solutions to the original problem.

## 7.2 FINDING FEASIBLE SOLUTIONS

When the algorithm terminates due to condition 1, one knows which constraints were violated by the solution to the best surrogate. Depending on the format (e.g., Balas) and type of problem considered (e.g., set covering), one may be able to raise additional variables to the one level in order to satisfy the violated constraints. The last set of knapsack ratios can be used to determine the order of raising additional variables to the one level. For the 18 problems studied, it is possible to quickly find a feasible solution to the original problem. To get an indication of the effect of a feasible solution on the total computation time, the algorithm employing a dual multiplier surrogate every eight iterations carrying forward a maximum of four surrogates (2.3) was again used. The only difference was than an initial solution to use as a bound was provided. The 18 problems tested requires 38.62 seconds to resolve without the bounds and 19.97 seconds with the bounds. A total of .26 seconds of additional time is required to find feasible solutions for those problems in which a feasible solution is not determined while iterating to a best surrogate. The results are summarized in Table 1.

## 7.3 HISTORY OF VARIABLES AND CONSTRAINTS

When iterating to a best surrogate, one may use index sets to record which variables are at the one level in the solutions to the various surrogates. This information can be used to attempt to find a feasible solution or for developing

a heuristic for branching in the Balas Algorithm. If additional information on the variables is desired, the exact solution to the LP relaxation of (3) is immediately available once the knapsack ratios have been determined [Dantzig, 1957].

Similarly, one can use an index set to record which of the original constraints are violated while iterating to a best surrogate. This information can be used advantageously in the Balas Algorithm.

## 8. CONCLUSIONS

Using the heuristic procedure developed in this paper it is possible to generate surrogates of strength comparable to the dual multiplier surrogate in less than 10% of the time required to form the dual multiplier surrogate. Because of the way the surrogates are formed, one would expect that the time required to converge to a best surrogate would behave well as the size of the problem increases. The results of the experimentation conducted suggest the following research:

1. Develop an algorithm which employs the heuristically generated surrogate in a manner analogous to the dual multiplier surrogate.
2. Develop heuristics for exploiting the ancillary information developed when iterating to a best surrogate.
3. Adapt the procedure to the general integer problem.

## 9. ACKNOWLEDGMENTS

I am deeply indebted to Professor Fred Glover, University of Colorado, and Professor Hamdy Taha, University of Arkansas, for contributing essential ideas throughout the conduct of this research. I would also like to thank Professor Frank Grange, Colorado School of Mines, who provided the problem set and Professor Gerald Brown, Naval Postgraduate School, for his valuable suggestions and assistance in presenting the results.

TABLE 1

EFFECT OF A DUAL MULTIPLIER SURROGATE AND AN INITIAL STARTING SOLUTION ON THE SOLUTION TIMES OF A BALAS ALGORITHM

| a | b | c | d | e | f | g | h | i | j |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 6 | 5 | 537 | 370 | 0.10 | 0.12 | 0.11 | - | 368 |
| 2 | 6 | 10 | 4,134 | 3,800 | 0.14 | 0.19 | 0.14 | 0.01 | 3,700 |
| 3 | 6 | 4 | 1,882 | 1,800 | 0.07 | 0.12 | 0.08 | - | 1,800 |
| 4 | 8 | 2 | 2,772 | 2,600 | 0.07 | 0.07 | 0.09 | - | 2,600 |
| 5 | 10 | 10 | 98,960 | 98,500 | 0.28 | 0.43 | 0.36 | - | 83,369 |
| 6 | 14 | 3 | 37,407 | 35,777 | 0.56 | 0.32 | 0.19 | - | 35,777 |
| 7 | 15 | 10 | 4,127 | 4,015 | 0.76 | 0.99 | 0.89 | 0.01 | 3,245 |
| 8 | 20 | 10 | 6,155 | 6,120 | 6.28 | 0.83 | 0.59 | - | 6,010 |
| 9 | 25 | 2 | 167 | 148 | 0.35 | 0.39 | 0.37 | 0.01 | 112 |
| 10 | 28 | 10 | 12,462 | 12,400 | 107.26 | 3.57 | 1.11 | 0.03 | 12,150 |
| 11 | 28 | 2 | 142,019 | 141,278 | 5.68 | 2.39 | 0.43 | 0.02 | 139,508 |
| 12 | 28 | 2 | 131,637 | 130,883 | 11.14 | 3.85 | 0.49 | 0.04 | 129,773 |
| 13 | 28 | 2 | 99,647 | 95,677 | 9.58 | 2.91 | 1.57 | 0.01 | 83,868 |
| 14 | 28 | 2 | 122,505 | 119,337 | 1.13 | 0.77 | 1.08 | 0.02 | 104,689 |
| 15 | 28 | 2 | 100,433 | 98,796 | 15.97 | 4.13 | 0.40 | - | 98,796 |
| 16 | 28 | 2 | 131,355 | 130,623 | 12.68 | 2.13 | 0.44 | 0.03 | 129,723 |
| 17 | 39 | 5 | 10,672 | 10,618 | 380.81 | 8.94 | 5.29 | 0.04 | 10,077 |
| 18 | 50 | 5 | 17,007 | 16,537 | (>5631) | 6.47 | 6.34 | 0.04 | 12,753 |
| SUM | | | | | (>6183) | 38.62 | 19.97 | 0.26 | |

a = Problem number

b = Number of variables

c = Number of constraints

d = Optimum solution the the continuous relaxation of the problem

e = Integer optimum solution

f = Solution time (CPU seconds), Balas Algorithm without any surrogates

g = Solution time (CPU seconds), Balas Algorithm with dual multiplier surrogate

h = Solution time (CPU seconds), Balas Algorithm with dual multiplier surrogate and a feasible initial starting solution

i = Additional time required to find a feasible solution if one had not been found after iterating to best surrogate

j = Feasible initial starting solution found

TABLE 2

COMPARISON OF THE DUAL MULTIPLIER AND HEURISTIC SURROGATES

| a | b | c | d | e | f | g | h | i | j |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 5 | 167 | 5 | 100 | 100 | 100 | 0.12 | 0.01 |
| 2 | 6 | 10 | 334 | 100 | 100 | 100 | 100 | 0.21 | 0.04 |
| 3 | 6 | 4 | 82 | 100 | 100 | 100 | 100 | 0.14 | 0.02 |
| 4 | 8 | 2 | 172 | 42 | 100 | 100 | 100 | 0.12 | 0.01 |
| 5 | 10 | 10 | 46 | 100 | 100 | 100 | 100 | 0.36 | 0.01 |
| 6 | 14 | 3 | 1,630 | 100 | 100 | 100 | 100 | 0.23 | 0.01 |
| 7 | 15 | 10 | 112 | 20 | 65 | 65 | 65 | 0.49 | 0.07 |
| 8 | 20 | 10 | 35 | 100 | 100 | 100 | 100 | 0.68 | 0.05 |
| 9 | 25 | 2 | 19 | 16 | 74 | 58 | * | 0.34 | 0.05 |
| 10 | 28 | 10 | 62 | 35 | 35 | ** | ** | 0.98 | 0.10 |
| 11 | 28 | 2 | 741 | 64 | 100 | 64 | 64 | 0.41 | 0.05 |
| 12 | 28 | 2 | 754 | 100 | 100 | 100 | 100 | 0.40 | 0.04 |
| 13 | 28 | 2 | 3,970 | 44 | 44 | 22 | 22 | 0.38 | 0.07 |
| 14 | 28 | 2 | 3,168 | 45 | 59 | 45 | 45 | 0.43 | 0.05 |
| 15 | 28 | 2 | 1,637 | 100 | 100 | 100 | 100 | 0.36 | 0.02 |
| 16 | 28 | 2 | 732 | 85 | 100 | 100 | 85 | 0.39 | 0.06 |
| 17 | 39 | 5 | 54 | 19 | 20 | 17 | *** | 0.96 | 0.12 |
| 18 | 50 | 5 | 47 | 19 | 87 | 87 | 38 | 1.17 | 0.11 |
| SUM | | | | | | | | 9.19 | 0.89 |

a. Problem number

b. Number of variables

c. Number of constraints

d. Absolute value of the difference between objective function values of the continuous and integer optimal solutions

e. Percent convergence of the dual multiplier surrogate

f. Percent convergence of best surrogate obtained heuristically

g. Percent convergence of best surrogate obtained incorporating Method 2 for forming the initial surrogate

h. Percent convergence of best surrogate obtained incorporating Method 2 for forming the initial surrogate and an approximate knapsack algorithm

i. Time required to form the dual multiplier surrogate (CPU seconds)

j. Time required to form surrogate in column h (CPU seconds)

---

*Solution to best surrogate was 191 while solution to dual multiplier surrogate was 164.

**Solution to best surrogate was 12,470 while solution to dual multiplier surrogate was 12,440.

***Solution to best surrogate was 10,774 while solution to dual multiplier surrogate was 10,662.

BIBLIOGRAPHY


1. Balas E. "An Additive Algorithm for Solving Linear Programs With Zero-One Variables", Operations Research, 13. (1965), 517-546.

2. Balas, E. "Discrete Programming by the Filter Method", Operations Research, 15. (1967), 915-957.

3. Dantzig, G.B. "Discrete Variable Extremum Problems", Operations Research, 5. (1957), 266-277.

4. Garfinkel, R.S., and G.L. Nemhauser. Integer Programming. Wiley, 1972.

5. Geoffrion, A.M. "An Improved Implicit Enumeration Approach for Integer Programming", Operations Research, 17. (1969), 437-454.

6. Giordano, F. "The Strength of Surrogate Constraints for the Linear Zero-One Integer Programming Problem", Naval Postgraduate School Technical Report, NPS55-082-008, February 1982.

7. Glover, F. "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem", Operations Research, 13. (1965), 879-919.

8. Glover, F. "Surrogate Constraints", Operations Research, 16. (1968), 741-749.

9. Taha, H.A. Integer Programming: Theory, Applications, and Computations. Academic Press, 1975.

DISTRIBUTION LIST

| | NO. OF COPIES |
|---|---|
| Library, Code 0142<br>Naval Postgraduate School<br>Monterey, CA 93940 | 4 |
| Dean of Research<br>Code 012A<br>Naval Postgraduate School<br>Monterey, CA 93940 | 1 |
| Library, Code 55<br>Naval Postgraduate School<br>Monterey, CA 93940 | 2 |
| Professor F. R. Giordano<br>Code 53Gi<br>Naval Postgraduate School<br>Monterey, CA 93940 | 48 |
| Defense Technical Information Center<br>ATTN: DTIC-DDR<br>Cameron Station<br>Alexandria, Virginia 22314 | |