Surrogate Constraints in Integer Programming

Bezalel Gavish
Owen Graduate School of Management
Vanderbilt University
Nashville, Tennessee 37203
U.S.A.

Fred Glover
Graduate School of Business Administration
University of Colorado at Boulder
Campus Box 419
Boulder, Colorado 80309
U.S.A.

Hasan Pirkul
College of Business
The Ohio State University
1775 College Road
Columbus, Ohio 43210
U.S.A.

ABSTRACT

We present an improved algorithm for generating surrogate constraint multipliers that constitues a hybrid approach based on integrating different procedures proposed earlier by Glover and by Gavish and Pirkul. We demonstrate the validity of the new procedure, give a numerical illustration, and report computational outcomes showing the effectiveness of the procedure in generating surrogate constraints for 240 multiple constraint knapsack problems drawn from an earlier study by Gavish and Pirkul.

1. INTRODUCTION

The surrogate constraint concept was first introduced by Glover (1965). A recent study by Gavish and Pirkul (1985) employs the surrogate constraint approach to develop an effective solution method for solving multiconstraint zero-one knapsack problems. Both papers propose methods for creating surrogate constraints from multiple parent constraints based on special procedures to generate surrogate constraints from exactly

Journal of Information & Optimization Sciences
Vol. 12 (1991), No. 2, pp. 219—228
©Analytic Publishing Co.

two parent constraints. The Glover procedure identifies progressive exact lower bounds for incrementing the multiplier of a violated parent constraint while the Gavish and Pirkul procedure fixes one multiplier equal to one and uses bisection to move brackets on the other multiplier until an ϵ-neighbourhood of an optimal value for this multiplier is determined.

Combining the perspectives of (Gavish and Pirkul 1985) and (Glover 1965) leads to a surrogate constraint generating procedure with interesting properties. The resulting procedure is superior to the original algorithms in a number of ways. The Gavish and Pirkul procedure does not have a mechanism to recognize an optimal multiplier when such a multiplier is obtained. The new algorithm, in addition to finding an ϵ-neighbourhood of an optimal multiplier, has the capability of recognizing an optimal multiplier as such. In almost all problems we solved to test the algorithm an optimal multiplier was identified. Furthermore the new procedure is able to recognize a range over which the bound will not improve and move the cuts further than a simple bisection algorithm would do and therefore improve the efficiency significantly. The algorithm outlined here also solves the potential problems that may be caused by using too large or too small step sizes in the original procedure proposed by Glover (1965).

In section 2 we outline basic surrogate constraint results. The new procedure is outlined in section 3. Section 4 presents a numerical example demonstrating the procedure. Computational results are presented in section 5.

2. BASIC SURROGATE CONSTRAINT IDEAS

Consider the following standard $0-1$ integer programming problem :

Problem IP

$$Z_{IP}=\text{Max}\ \{cx \mid Ax\leqslant b,\ x\in\{0,1\}\}.$$

The surrogate relaxation of this problem is given by :

Problem S

$$Z_S(\mu)=\text{Max}\ \{cx \mid \mu(Ax-b)\leqslant 0,\ x\in\{0,1\}\}.$$

Here we concentrate on the two constraint version of the problem. Since procedures outlined both in (Glover 1965) and in (Gavish and Pirkul 1985) to generate surrogate multipliers for the general problem use a series of two constraint problems, the results derived here will have implications for the general problem as well.

In (Glover 1965) a range of surrogate multipliers over which the solution of the surrogate problem remains unchanged is determined. To present this result we use the following simple framework. Let S denote a surrogate constraint corresponding to a multiplier vector μ and let x^* denote the solution to problem S with multiplier vector μ. Taking a slight liberty with notation, suppose S is divided into two component constraints F and G such that $S=F+G$. Moreover suppose F is satisfied by x^* and G is not. Finally, let f and g be the (scalar) amounts by which x^* over-satisfies F and undersatisfies G. Then it is shown in (Glover 1965) that the surrogate constraint represented by $F+kG$ will be satisfied by x^* for $0\leqslant k\leqslant f/g$, and will fail to be satisfied when $k=f/g+\epsilon$, for all $\epsilon>0$. It is also shown in (Glover 1965) that at most one of the constraints can be violated by x^*, and that the value $Z_S(\mu)$ decreases monotonically (non-strictly) as the weight on the violated constraint is increased, until the constraint is no longer violated. As observed by Greenberg and Pierskalla (1970) this implies $Z_S(\mu)$ is quasi-convex, and as further noted in Gavish and Pirkul (1985) :

1. If we let μ' be a multiplier value such that solution of $Z_S(1, \mu'+\epsilon)$ satisfies the second constraint and the solution of $Z_S(1, \mu'-\epsilon)$ violates the second constraint for any $\epsilon>0$, then for μ_1, μ_2 such that
$$0<\mu_1<\mu_2<\mu',\ Z_S(1, \mu_1)\geqslant Z_S(1, \mu_2).$$
2. If we let μ'' be a multiplier value such that solution of $Z_S(1, \mu''+\epsilon)$ violates the first constraint and the solution of $Z_S(1, \mu''-\epsilon)$ satisfies the first constraint for any $\epsilon>0$, then for μ_1, μ_2 such that
$$\mu''<\mu_1<\mu_2<\infty,\ Z_S(1, \mu_1)\leqslant Z_S(1, \mu_2).$$

The preceding relationships were used in (Gavish and Pirkul 1985) to propose an algorithm which keeps the multiplier of the first constraint equal to 1 and starts with two multiplier values μ_l and μ_h bracketing the optimal multiplier range for the second constraint. This range is then reduced through a bisection procedure. The algorithm continues bisection cuts until the difference between the two brackets is less than or equal to a prespecified constant ϵ.

We now present a hybrid procedure that integrates the ideas underlying these two different surrogate constraint generation approaches.

3. THE IMPROVED PROCEDURE FOR GENERATING SURROGATE CONSTRAINTS

Procedure-GGP

1. Let μ_h and μ_l be two multiplier values such that the solution of $Z_S(1, \mu)$ satisfies the first constraint at $\mu=\mu_l$ and does not satisfy it at $\mu=\mu_h$.

2. Let $\bar{\mu}=\mu_l+(\mu_h-\mu_l)/2$. Solve $Z_S(1, \bar{\mu})$. If in the solution to $Z_S(1, \bar{\mu})$,

   (*i*) both constraints are satisfied STOP, $(1, \bar{\mu})$ is an optimal multiplier vector and an optimal solution to the original problem (problem-IP) has been obtained.

   (*ii*) only the first constraint is satisfied, let $\mu_l=f/g$ where f is the amount of oversatisfaction of the first constraint and g is the amount of undersatisfaction of the second constraint.

   (*iii*) only the second constraint is satisfied, let $\mu_h=f/g$ where f is the amount of undersatisfaction of the first constaint and g is the amount of oversatisfaction of the second constraint.

3. If $\mu_h-\mu_l \geqslant \epsilon$ GO TO STEP 2, otherwise STOP. If $\mu_l \geqslant \mu_h$, $(1, \bar{\mu})$ is an optimal multiplier vector. Otherwise both μ_l and μ_h are within an ϵ-neighbourhood of an optimal multiplier for the second constraint ; choose the one that corresponds to the lowest bound as the final multiplier.

It is evident that the foregoing process will either obtain an optimal solution at Step 2 (*i*) or the difference $\mu_h-\mu_l$ will be diminished by at least half each time Step 2 is visited. (The ratio assignments of Steps 2(*ii*) and 2(*iii*) always move in the direction of diminishing the difference and the effect is compounded by the bisection that determines $\bar{\mu}$.) The essential requirement is therefore to justify the assertions of Step 3 when the algorithm terminates.

THEOREM. *For μ_l and μ_h as determined by the algorithm, whenever $\mu_h - \mu_l < \epsilon$; the condition $\mu_l \geqslant \mu_h$ implies that $(1, \bar{\mu})$ is an optimal multiplier vector, and otherwise μ_l and μ_h are within an ϵ-neighbourhood of optimality.*

PROOF. The result derives from the observations concerning the f/g ratio outlined in section 2. There are two cases to consider.

Case 1. Only the first constraint is satisfied.

Let the multipliers be 1 and $\bar{\mu}$ for the first and second constraints respectively. Therefore F is equal to the first constraint and G is equal to the second constraint multiplied by $\bar{\mu}$.

$\Rightarrow$ f/g=(oversatisfaction of the first constraint)/$(\bar{\mu})$ (undersatisfaction of the second constraint)

$\Rightarrow$ $F+KG$=first constraint$+(f/g)$ $(\bar{\mu})$ (second constraint)

$\Rightarrow$ Multiplier of the second constraint$=(f/g)$ $(\bar{\mu})$.

Substituting for f/g from above, the multiplier of the second constraint becomes equal to oversatisfaction of the first constraint divided by the undersatisfaction of the second constraint. Since the solution corresponding to multiplier vector $(1, \bar{\mu})$ will satisfy the surrogate constraint until μ reaches this ratio, the lower bracket (μ_l) can be set equal to the ratio rather than $\bar{\mu}$.

Case 2. Only the second constraint is satisfied.

Let multipliers be 1 and $\bar{\mu}$ for the first and second constraints respectively. Therefore in this case F is equal to the second constraint multiplied with $\bar{\mu}$ and G is equal to the first constraint

$\Rightarrow$ f/g= $(\bar{\mu})$ (oversatisfaction of the second constraint)/undersatisfaction of the first constraint

$\Rightarrow$ $F+kG=(\bar{\mu})$ (second constraint$+(f/g)$ (first constraint).

Normalizing the multipliers so that the multiplier of the first constraint is equal to 1 ;

Multiplier of the second constraint$=(\bar{\mu}\, g)/f$.

Substituting for g/f from above, the multiplier of the second constraint becomes equal to the undersatisfaction of the first constraint divided by the oversatisfaction of the second constraint. Since the solution corresponding to multiplier vector $(1, \bar{\mu})$ will satisfy the surrogate constraint until μ reaches this ratio, the upper bracket (μ_h) can be set equal to the ratio rather than $\bar{\mu}$. This completes the proof. □

We next provide a numerical example and then report computational results.

4. NUMERICAL EXAMPLE

The algorithm is demonstrated on the following simple problem from (Gavish and Pirkul 1985) :

$c=\{45\ 57\ 43\ 44\ 61\ 35\ 41\ 54\ 18\ 22\ 78\}$

$$A=\begin{Bmatrix} 94 & 24 & 69 & 95 & 63 & 2 & 98 & 29 & 15 & 9 & 100 \\ 26 & 91 & 35 & 15 & 80 & 71 & 20 & 76 & 33 & 46 & 86 \end{Bmatrix}$$

$b=\{351\ 192\}$.

The second constraint is the tighter of the two constraints so we fix the multiplier of the first constraint equal to 1 and search for a multiplier for the second constraint. Figure 1 illustrates $S(1, \mu)$ as a function of μ.

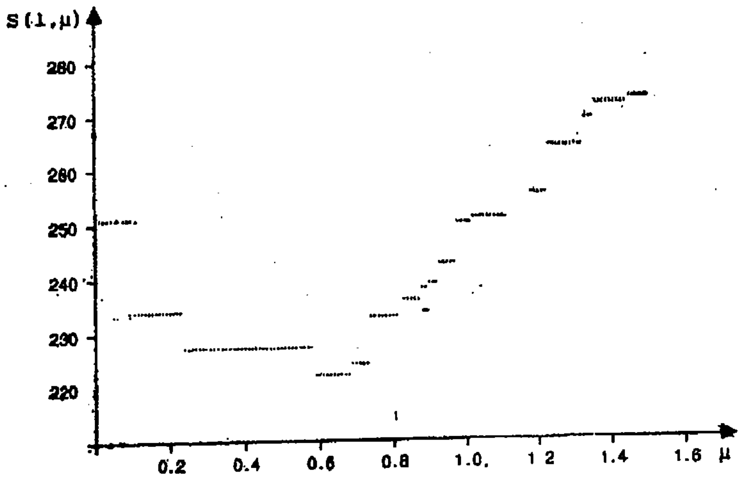

Figure 1. The function of $S(1, \mu)$ plotted as a function of μ.

The cuts made by procedure-GGP to reach an optimal multiplier are illustrated in Table 1. After the fifth iteration the upper bracket is moved below the lower bracket indicating that (1, .6041) is an optimal multiplier. Procedure 1 from (Gavish and Pirkul 1985) would not be able to confirm an optimal multiplier. Furthermore it would take 8 cuts for an ϵ value of .01. The number of cuts would go up to 12 for an ϵ of .001.

TABLE 1

Bisection cuts in Procedure GGP

Iter.	μ_l	μ_h	*Interval size*	*Objective function*	*Constraint violated*	*Bisection cut*
1	0.	—	1.0000	250	1	1.0000
2	0.	.9817	.9817	227	2	.4909
3	.5882	.9817	.3935	233	1	.7850
4	.5882	.7273	.1391	222	1	.6578
5	.5882	.6200	.0318	222	1	.6041
6	.5882	.5556	−.0326			

5. COMPUTATIONAL RESULTS

We have tested procedure GGP on a wide range of multiple constraint knapsack problems drawn from the study of (Gavish and Pirkul 1985) to provide the basis of comparison. The program for procedure GGP as for Procedure 1 in (Gavish and Pirkul 1985) was written in FORTRAN-77, and the computational experiments were conducted on a PRIME 9955 minicomputer. The results of the comparative tests are summarized in Tables 2—4. A total of 240 problems arranged into groups of 10 problems were solved. The results with 150 of these problems are reported in Table 2. For these problems, the objective function coefficients (c_i) as well as the resource consumption coefficients (a_{ij}) defining constraints were drawn from uniform distributions and were rounded up to the nearest integer value. The right hand side element (b_j) for each constraint

TABLE 2*

Comparison of procedure GGP and procedure 1

c_i and a_{ij} are drawn from :	*Number of Variables*	*Average Number of Cuts with Procedure 1*	*Average Number of Cuts with Procedure GGP*	*% of Cases Opt. Mult. Confirmed*
	10	8.6	2.1	100
	50	10.6	3.0	100
U(1, 10)	100	10.8	3.3	100
	200	11.0	3.9	100
	300	10.5	3.6	90
	10	10.4	2.1	100
	50	10.8	3.9	100
U(1, 100)	100	10.4	3.9	100
	200	11.3	3.7	90
	300	11.2	5.9	70
	10	8.7	2.7	100
	50	11.2	3.7	100
U(1, 1000)	100	10.8	5.6	100
	200	11.3	4.8	100
	300	10.6	5.7	100

(*) An ϵ value of .001 was used in both procedures.

TABLE 3*

Performance of procedures as the right hand side element values are reduced

The Right Hand Side Element b_j equal to :	*Average Number of Cuts with Procedure 1*	*Average Number of Cuts with Procedure GGP*	*% of Cases Opt. Mult. Confirmed*
$\Sigma a_{ij}/2$	10.4	3.9	100
$\Sigma a_{ij}/4$	10.8	4.1	100
$\Sigma a_{ij}/6$	11.0	4.2	100
$\Sigma a_{ij}/8$	11.0	4.4	100
$\Sigma a_{ij}/10$	11.3	4.5	100

(*) An ϵ value of .001 was used in both procedures. Problems in all groups consisted of 100 variables.

TABLE 4*

Performance of the procedures on problems where objective function coefficients are correlated to the resource consumption coefficients

The Multiplier K used to define c_j equal to:	*Average Number of Cuts with Procedure 1*	*Average Number of Cuts with Procedure GGP*	*% of Cases Opt. Mult. Confirmed*
50	11.7	8.0	100
100	11.4	7.5	80
300	11.2	6.0	80
600	11.3	6.2	100
1000	11.3	5.7	100

(*) An ϵ value of .001 was used in both procedures. Problems in all groups consisted of 100 variables.

was set equal to half of the sum the resource consumption coefficients (a_{ij}). For each group of problems, Table 2 reports the average number of cuts using procedure GGP as well as procedure 1. Also reported is the percent of cases where an optimal multiplier is confirmed by procedure GGP. As can be seen from this table, procedure GGP requires on the average around 63 percent fewer cuts than Procedure 1. Furthermore in 145 of the 150 problems an optimal multiplier was confirmed with this procedure. It should also be pointed out that the relative savings will increase as we use smaller ϵ values in these procedures.

The procedures are further compared on two different sets of problems. The first set is obtained by reducing the right hand side element of constraints thus forming problems in which fewer and fewer variables can be fixed to 1. The objective function and the resource consumption coefficients of these problems were both drawn from a uniform distribution between 1 and 100 and were rounded up to the nearest integer number. Problems in all groups consisted of 100 variables. The results with these problems which are reported in Table 3 are similar to the results reported in Table 2. The second problem set was generated to compare the algorithms on problems where the objective function coefficients (c_i) are correlated with the resource consumption coefficients (a_{ij}). Five groups of problems were generated. All problems in these groups consisted of 100 variables. For these problems resource consumption coefficients were drawn from a uniform distribution between 1 and

1000 and rounded up to the nearest integer values. The objective function coefficients were generated as follows :

$$c_i = [(a_{i1} + a_{i2})/2 + K^*r],$$

where $[d]$ represents the smallest integer greater than d, r is random number from a uniform distribution between 0 and 1 and K is a multiplier. This multiplier is used to generate problems with varying degrees of correlation between c_i and a_{ij}. The results are reported in Table 4. It is observed that, for these problems the savings in the number of cuts is around 41 per cent which is somewhat smaller than the savings with other sets.

REFERENCES

1. B. Gavish and H. Pirkul, Efficient Algorithms for Solving Multiconstraint Zero-One Knapsack Problems to Optimality, *Mathematical Programming*, Vol. 31 (1985), pp. 78-105.

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

3. H. J. Greenberg and W. P. Pierskalla, Surrogate Mathematical Programs, *Operations Research*, Vol. 18 (1970), pp. 924-939.

Received January, 1990