UNIVERSIDADE NOVA DE LISBOA
Faculdade de Economia

"A GENERALISED BOUND IMPROVEMENT SEQUENCE ALGORITHM"

Paulo Bārcia e Soren Holm

Working Paper nº 54

UNIVERSIDADE NOVA DE LISBOA
Faculdade de Economia
Campo Grande, 185
1700 LISBOA

Outubro 1986

A GENERALISED BOUND IMPROVEMENT SEQUENCE ALGORITHM

by

PAULO BÂRCIA (*)

and

SOREN HOLM (**)

ABSTRACT

In this paper we present a generalization and a computational improvement of the Bound Improvement Sequence Algorithm.

The main computational burden of this algorithm consists in determining whether there exists a feasible point on the objective hyperplane, when the algorithm encounters a fixed point.

By generalizing the algorithm, which consists in treating the objective function and the constraints alike, the number of fixed points for the objective hyperplane can be reduced, thus making the algorithm more efficient.

We give computational results comparing the original algorithm with the proposed generalized algorithm, wich shows that, for loosely constrained problems, the number of fixed points can generaly be reduced.

(*) Universidade Nova de Lisboa, Faculdade de Economia,
Campo Grande 185, 1700 Lisboa, Portugal

(**) Odense University, Departement of Management,
Campusvej 55, 5230 Odense, Danmark

1 - INTRODUCTION

Recently a new method for solving general pure Integer Programming problems has been suggested (Bárcia 85, 86).

The algorithm is based upon the construction of progressively tighter dual problems. If a particular dual fails to solve the primal problem a stronger dual is built. The algorithm continues until the primal is solved and it can be shown that this occurs in a finite number of iterations.

The method has, however, a drawback. The constrution of a new dual is in general obtained by the solution of a sequence of knapsack problems, but sometimes this is not enough to produce a new dual, so a subroutine consisting in enumeration has to be performed, thereby deteriorating the performance of the algorithm.

In this paper we will provide a characterization of the situations which require enumeration and use it to construct an algorithm which requires a lesser amount of enumeration work. Is turns out that this new algorithm is a generalization of the basic principles underlying the original one.

The plan of this paper is the following:

In section 2 we shall recall the basics of constructive dual methods in integer programming and give a description of the Bound Improving Sequence Algorithm (BISA). In section 3 we shall provide a characterization of the situations under which BISA requires an enumeration subroutine and we will use those results to suggest a more general BISA requiering less enumeration. Section 4 will be devoted to computational experience comparing the old and new versions of the algorithm. Finaly in section 5 we shall draw some conclusions and point out the class of problems for wich the generalised method appears to be useful.

2 - THE BOUND IMPROVING SEQUENCE ALGORITHM

In this section we shall briefly sketch the Bound Improving Sequence Algorithm (BISA). We shall deal only with the main ideas and we will not provide proofs for the results stated. The reader is referred to (Barcia 85, 86) for a full description and proofs.

Consider an integer programming problem of the form

$$\begin{aligned} z &= \min cx \\ (IP) \qquad & Ax \geq b \\ & x \in X \end{aligned}$$

where c is in Z^n, b is in Z^m, A is a mxn matriz of integers and X is a bounded subset of Z^n. Suppose further that an optimal solution exists and let z be the value of such a solution.

Now let $s(k) < z$ be available. Then the following problem is equivalent to IP

$$\begin{aligned} z &= \min cx \\ & Ax \geq b \\ & cx \geq \lceil s(k) \rceil \\ & x \in X \end{aligned}$$

where $\lceil y \rceil$ stands for the integer imediately after y, i.e., $\lceil 2.5 \rceil = 3 = \lceil 2 \rceil$

If we take a (Lagrangean) dual of the above problem we will have

$$\begin{aligned} s(k+1) &= \max \quad \min \quad cx + u\,(b-Ax) \\ (BIS) \qquad & u \geq 0 \qquad cx \geq \lceil s(k) \rceil \\ & \qquad\quad x \in X \end{aligned}$$

It can be shown that we will always have the following inequalities:

$$s(k) < s(k+1) \leq z .$$

so we have generated a bound improving sequence (BIS).

Now, if we have a way of testing if a particular s(k) is equal to z BIS will define an algorithm (BISA) which will bridge the duality gap in a finite number of steps. The following theorem gives a necessary and sufficient condition for s(k)=z. Of course, one can only have s(k) = z if s(k) is an integer.

THEOREM: Consider an integer $s \leq z$ wich is a fixed point for

$$\begin{aligned} s = \max_{u \geq 0} \min_{\substack{cx \geq s \\ x \in X}} & cx + u(b - Ax) \end{aligned}$$

Then s = z if and only if there exists an optimal solution for the above fixed point problem of the form (x, u = 0) such that $Ax \geq b$.

Using the result stated above we can outline an algorithm to solve IP as follows: start with any s(k) < z and use BIS while no fixed point having u = 0 as the optimal multiplier occurs. When we have such a fixed point we must enumerate, searching X for a point on the hyperplane cx = s(k) such that $Ax \geq b$. If such a point exists it is an optimal solution for IP and we have s(k) = z. If no such point exists then s(k) < z and we can restart BIS.

We now make a formal statment of the algorithm:

```
begin procedure BISA
    k ← 0
    optimal ← false
    while (not optimal) loop
          s(k+1) ← max  min  cx + u (b-Ax)
                   u ≥ 0      cx ≥ ⌈s(k)⌉
                              x ε X
          if (s(k+1) = ⌈s(k)⌉ and  u=0 is optimal) then
              enumerate on cx = s(k+1), x ε X, to find
              a feasible x, Ax ≥ b.
              if (x was found) then
                  x is optimal for IP
                  optimal ← true
              end if
          else
                k  ← k + 1
          end if
    end loop
end BISA
```

Of course, the agorithm will be more efficient if only a small number of fixed points are encountered. In fact the main computational burden lies in the search for a feasible point (or to prove that none exists) on the hyperplane $cx = s(k+1)$, which has to be performed using some fast enumeration scheme. The BIS formulation poses no such problem because the inner minimization problem is a knapsack problem for which efficient algorithms are available, see (Martello and Toth 86) , and the outer maximization problem can be performed using a subgradient type algorithm, see (Mahey 86), since only an approximate solution is needed (only $\lceil s(k) \rceil$ is important

for the next iteration, not s(k)).

In the next section we will suggest a way of reducing the number of fixed points encountered by the BISA algorithm. But before terminating this section we shall take a slightly different look at the algorithm which will be useful later on.

Take the hyperplane $cx = s(k) < z$ and a set of constraints defined as $T = \{x \in X : Ax \geq b\}$. The BISA algorithm "pushes" the objective hyperplane $cx = s(k)$ until a feasible point $x \in T$ is encountered and we have perforce $cx=z$. Of course, one could push any hyperplane defining a valid cut for T, so the above algorithm can be viewed as a "method for strenghtning any valid cut for T until it reaches a feasible, but not necessary optimal, integer point".

4 - WHY ENUMERATION IS NECESSARY

In the last section we saw that the BISA requires enumeration when it generates a lower bound on z, s(k), such that s(k) is an integer and we have that u=0 is an optimal multiplyer for the following fixed point problem

$$s(k) = \max_{u \geq 0} \min_{cx \geq s(k),\ x \in X} cx + u(b - Ax)$$

It is known, see for instance (Barcia 86), that solving the above Lagragean dual provides the same optimal value as solving the following convexified primal

$$s(k) = \min cx$$
$$Ax \geq b$$
$$x \in \text{Conv}\ \{x \in X: cx \geq s(k)\}$$

A fixed point is only encountered when the equality in the convexified primal problem is obtained, and this only happens if then exists an optimal solution x^*, not necessarely integer, to the problem and at least one integer point for which $cx = s(k)$. If either x^* is an integer point or any of the integer points x are feasible, $Ax \geq b$, then it is optimal. So each time we encounter a fixed point we have to determine whether there exists an integer feasible point on the hyperplane $cx = s(k)$. This is done by enumeration and computational experiences show that we may encounter many fixed points before we reach the optimal solution.

From the theoretical argument presented above, where we gave the conditions for a fixed point on the basis of the convexified primal problem rather the Lagrangean dual which we are actually solving. A necessary condition for a fixed point to exist is, that there exists an optimal x^* such that $cx^* = s(k)$. Observe that $Ax^* \geq b$. Suppose now that x^* is not an integer point. If we were to change b such that $Ax^* \not\geq b$ then the fixed point would not be encountered. Of course, the change in b would have to be valid for the problem.

This suggests that when a fixed point is encountered for the objective row, one should try to push a constraint instead of performing an enumeration, hoping to eliminate the fixed point.

If the previons section we argued that BISA can be viewed as a method which pushes any hyperplane until it lies on a feasible integer point.

Of course one should only try to push a constraint so long as no fixed point occurs for that particular hyperplane. In fact one could push all hyperplanes (constraints and objective row) in sucression, hoping that pushing a hyperplane would destroy some of the fixed points previously encountered by the others.

We now present a generalization of BISA which makes use of this idea. For the sake of simplicity we shall consider an IP of the form.

$$
\begin{aligned}
z = \min\ & a(0)x \\
& a(0)x \geq b(0) \\
& a(1)x \geq b(1) \\
& \\
& a(m)x \geq b(m) \\
& x \in X
\end{aligned}
$$

Where b(0) is a lower bound on z. The matrix A will stand for a matrix with rows a(0), a(1), ..., a(m) and the vector b for an (m+1)-vector of components b(0), b(1), ..., b(m).

```
begin procedure GBISA
    terminate  ← false
    enumerate  ← false
    while (not terminate) loop
        while (not enumerate) loop
            enumerate  ← true
            for j = 0, ..., m loop
                c ← a(j)
                s(0) ← b(j)
                solve min {cx : Ax ≥ b; x ∈ X}  using
                BISA until one gets a fixed point s
                if(s > s(0)) then
                    enumerate ← false
                    b(j) ← s
                end if
            end loop
        end loop
        enumerate on a(0)x = b(0), x ∈ X
        if (a feasible x (Ax ≥ b) was found) then
            x is optimal
            terminate ← true
        else
            b(0) ← b(0) + 1
        end if
    end loop
end GBISA
```

This procedure pushes any hyperplane (constraint or objective row) as long as fixed points are not encountered by any hyperplane.

The method can be further generalized by including generated facets and valid cuts, see (Barcia and Holm 86).

Only when every hyperplane is bloked on a fixed point is the enumeration procedure used.

Of course, one could have searched any hyperplane for feasible points but since these may not be optimal we restrict enumeration to the objective row.

4 - COMPUTATIONAL EXPERIENCE

In this section we shall give computational experience with the GBISA algorithm and compare it with the BISA performances for the case of 0-1 linear programming problems.

We have generated 9 random problems with 20 constraints and 40 0-1 variables. The matrix coeficients and the cost row were uniformely random generated in $[0, 1000]$ The right hand side coeficients were obtained by summing up the corresponding constraint coeficients and dividing the result by 10. Thus we shall have loosely constrained problems in the sense that only about 1/10 of the variables will have the value 1 in the optimal solution.

Table 1 compares the performances of BISA and the generalised algorithm presented in this paper. It shows that a reduction of the number of fixed points and iterations is generaly obtained.

Table 1

Problem		BISA		GBISA	
nº	Gap	NIT	NFP	NIT	NFP
1	486	98	40	90	32
2	341	67	21	66	12
3	274	86	26	79	24
4	304	167	96	138	76
5	402	71	26	71	26
6	78	12	2	12	0
7	377	150	64	60	16
8	275	126	53	125	52
9	225	24	19	24	17

Gap: duality gap for the LP bound
NIT: number of iterations

We also tried to study the behaviour of the generalized algorithm on more tightly constrained problems, where the right hand side was obtained by dividing the sum of the corresponding row coeficients by 2.

Unfortunatly for those problems no gain over the usual BISA could be obtained. This is due to the fact that the constraints could seldomly be "pushed" by a meaningful amount, because they were already very "near" a feasible integer point.

5 - CONCLUSIONS

The main computational burden of the BISA type algorithms consists in determining whether there exists a feasible point on the objective hyperplane, when the algorithm encounters a fixed point.

By generalizing the algorithm, which consists in treating the objective function and constraints alike we may be able to reduce the number of fixed points encountered.

However computational experience shows that this gain is only meaningful for loosely constrained problems, i.e., for problems for which the right hand side is small if compaired to the sum of the coeficients of the corresponding row. This is generaly due to the fact that if the right hand side is "big" the constraint can not be pushed further because it is already "near" a feasible integer point.

6 - REFERENCES

Barcia, P. (1985), "The bound improving sequence algorithm", Operations Research Letters, vol. 4, nº 1, 27 - 30.

Barcia, P. (1986), "Constructive dual methods for discrete programming", Discrete Applied Mathematics, forthcoming.

Barcia, P.; Holm, S. (1986) "Improving cuts by constructive dual methods", paper presented at the EURO VIII meeting, Lisbon.

Mahey, P. (1986), "Subgradient techniques and combinatorial optimization", Annals of Discrete Mathematics, forthcoming

Martello, S.; Toth, P. (1986), "Algorithms for knapsack problems", Annals of Discrete Mathematics, forthcoming.