Discussion Paper No. 46

BOXSTEP: A NEW STRATEGY FOR LARGE SCALE MATHEMATICAL PROGRAMMING

by

W. W. Hogan*, R. E. Marsten**, and J. W. Blankenship***

March, 1973

(Revised May, 1973)

*U. S. Air Force Academy

** Northwestern University

*** Institute for Defense Analyses

Abstract

A new strategy is presented for large scale mathematical programming. Several specific applications are described and computational results are cited. These applications of the BOXSTEP strategy fall in the conceptual continuum between steepest ascent methods and outer approximation methods. BOXSTEP is able to capture the best features of both of these extremes while at the same time mitigating their bad features.

CONTENTS

1. Introduction

This paper presents a very simple idea that has unified some previously separate areas of theory and has produced some surprising computational advantages. The idea can be stated as follows. Suppose that we want to maximize a concave function $v(y)$ over a convex set Y. Let B denote a "box" (i.e., hyper-cube) for which $Y \cap B$ is non-empty. Let y^* be a point at which $v(y)$ achieves its maximum over $Y \cap B$. If y^* lies in the interior of the box, then by the concavity of v, y^* must be globally optimal. If, on the other hand, y^* lies on the boundary of the box, then we can translate B to obtain a new box B' centered at y^* and try again. By "try again" we mean maximize v over $Y \cap B'$ and check to see if the solution is in the interior of B'. This intuitive idea is developed rigorously in section 2. Note immediately that we presuppose some appropriate algorithm for solving each local problem. This "appropriate algorithm" is embedded in a larger iterative process, namely maximizing v over a finite sequence of boxes. Computational advantage can be derived if each local problem with feasible region $Y \cap B$ is significantly easier to solve than the global problem with feasible region Y.

The problems that we have in mind are those where $v(y)$ is the optimal value of a sub-problem (SPy) that is parameterized on y. Thus $v(y)$ is not explicitly available and evaluating it at $\hat{y}$ means solving (SP$\hat{y}$). This arises in the context of decomposition methods for large scale mathematical programs and it was in this context that the BOXSTEP idea was developed. We begin by presenting it in a more general context so as to facilitate its application to other kinds of problems, e.g., non-linear programs where $v(y)$ is explicitly available. In this case BOXSTEP bears some resemblance

to the Method of Approximation Programming (MAP) originally proposed by Griffith and Stewart [ ] and recently revived by Beale [ ] and Meyer [ ].

Section 2 presents the BOXSTEP method in very general terms and proves its convergence. Section 3 shows how an Outer Linearization/ Relaxation scheme can be used to solve each local problem. In this form BOXSTEP falls between the feasible directions methods at one extreme and outer approximation or cutting plane methods at the other extreme. One can obtain an algorithm of either type, or something "in between", by simply adjusting the size of the box. Sections 4, 5, and 6 contain specific applications to large structured linear programs. Section 7 relates BOXSTEP to other recent developments in large scale optimization and points out some very promising directions for additional research.

2. The BOXSTEP Method

BOXSTEP is not a completely specified procedure but rather a method of replacing a single difficult problem by a finite sequence of simpler problems. These simpler problems are to be solved by an appropriate algorithm. This "appropriate algorithm" may be highly dependent on problem structure but by assuming its existence and convergence we can establish the validity of the overall strategy. In this section, therefore, we present a general statement of the BOXSTEP method, prove its finite ϵ-optimal termination, and discuss some modifications of the basic method which will not upset the fundamental convergence property.

Consider any problem of the form

$$(P) \qquad \max_{y \in Y} \; v(y) \;, \quad \text{with } Y \subset R^n \text{ and } v: Y \to R \;.$$

If, for $\hat{y} \in Y$ and $\beta > 0$, the local problem

$$P(\hat{y};\, \beta) \qquad \max_{y \in Y} \; v(y) \quad \text{s.t.} \quad \|y - \hat{y}\|_\infty \leq \beta$$

is considerably easier to solve, either initially or in the context of a reoptimization, then (P) is a candidate for the BOXSTEP method.

BOXSTEP Method

Step 1: Choose $y^1 \in Y$, $\epsilon \geq 0$, $\beta > 0$. Let $t = 1$.

Step 2: Using an appropriate algorithm, obtain an ϵ-optimal solution of $P(y^t;\, \beta)$, the local problem at y^t. Let y^{t+1} denote this solution.

Step 3: If $v(y^{t+1}) \leq v(y^t) + \epsilon$, stop. Otherwise let $t = t + 1$ and go to Step 2.

The BOXSTEP mnemonic comes from the fact that at each execution of Step 2 the vector y is restricted not only to be in the set Y but also in

a box of size 2β centered at y^t and the box steps toward the solution as t is incremented. The appeal of this restriction springs both from heuristics and empirical observations which are discussed below. In essence, these results indicate that $\beta = +\infty$, which corresponds to solving problem (P) all at once, is not an optimal choice. Notice that the stopping condition at Step 3 is based on the objective function rather than on y^{t+1} being in the interior of the box. This is necessary because v(y) may be flat on top, in which case we might never obtain an interior solution.

The simplicity of the concept would indicate that the convergence of any algorithm is not upset when embedded in the BOXSTEP method. This is formally verified in the subsequent theorem for which we need the following definitions:

$$\delta \equiv \max_{x,y \in Y} \|x - y\|_2 \quad ,$$

$$\lambda \equiv \min \{\beta/\delta, 1\} \quad ,$$

and

$$v^* = \max_{y \in Y} v(y).$$

Theorem: If Y is a compact convex set and v is an upper semi-continuous concave function on Y, then the BOXSTEP method will terminate after a finite number of steps with a $2\epsilon/\lambda$ optimal solution.

Proof: First we establish that the method terminates finitely. If $\epsilon > 0$, then non-termination implies that $v(y^{t+1}) \geq v(y^1) + t\epsilon$ and, therefore, $\limsup v(y^t) = \infty$. This contradicts the fact that an upper semi-continuous function achieves its maximum on a compact set.

If $\epsilon = 0$, then either $\|y^t - y^{t+1}\|_\infty = \beta$ for each t or $\|y^T - y^{T+1}\|_\infty < \beta$ for some T. If $\|y^T - y^{T+1}\|_\infty < \beta$ then the norm constraints are not binding

and, by concavity, may be deleted. This implies that $v(y^{T+1}) = v^*$ and termination must occur on the next iteration. If $\|y^t - y^{t+1}\|_\infty = \beta$ for each t, then, without termination, $v(y^{t+1}) > v(y^t)$ for each t. If $\|y^s - y^t\|_\infty < \beta/2$ for any $s > t$, then this would contradict the construction of y^{t+1} as the maximum over the box centered at y^t (because $\epsilon = 0$). Therefore $\|y^s - y^t\|_\infty \geq \beta/2$ for all $s > t$ and for each t. This contradicts the compactness of Y. Hence the method must terminate finitely.

When termination occurs, say at step T, we have $v(y^{T+1}) \leq v(y^T) + \epsilon$. Let y^* be any point such that $v(y^*) = v^*$. Then, by concavity,

$$v\left((1-\lambda)y^T + \lambda y^*\right) \geq (1-\lambda)v(y^T) + \lambda v(y^*).$$

Now the definition of λ implies that

$$\|(1-\lambda)y^T + \lambda y^* - y^T\|_\infty \leq \beta ,$$

and by the construction of y^{T+1} it follows that

$$v(y^{T+1}) + \epsilon \geq v\left((1-\lambda)y^T + \lambda y^*\right) \geq (1-\lambda)v(y^T) + \lambda v^*.$$

Therefore, since termination occurred,

$$v(y^T) + 2\epsilon \geq (1-\lambda)v(y^T) + \lambda v^*$$

or

$$\lambda v(y^T) + 2\epsilon \geq \lambda v^*.$$

Hence,

$$v(y^T) \geq v^* - 2\epsilon/\lambda.$$

Q.E.D.

This robust convergence result requires at least one qualification. With the exception of the case where v is piecewise linear, there are not many situations in which it is clearly evident that Step 2 can be performed in a finite number of steps when $\epsilon = 0$ and the choice of $\epsilon > 0$ be-

comes mandatory. This touches upon the whole range of problems related to speed of convergence and numerical stability in the execution of Step 2 but, due to the intentional vagueness of the term "appropriate", it is not possible to discuss these except in the context of a specific application.

One important modification of the basic method that can be introduced without upsetting convergence is a line search. When far from the solution, it is possible to view BOXSTEP as an ascent method which uses Step 2 as a procedure employing more than strictly local information in the determination of the next direction of search, $d^t = y^{t+1} - y^t$. As stated above, a step of size one is taken in this direction. Depending on the structure of (P), it may be beneficial to do an exact or approximate line search maximizing $v\left(y^t + \vartheta(y^{t+1} - y^t)\right)$ over feasible values of $\vartheta \geq 1$. This can be incorporated in Step 3 in the natural way without disturbing the statement of the theorem.

It should be emphasized that the interpretation of BOXSTEP as an ascent method does not apply in the neighborhood of a solution. Near termination, the BOXSTEP method becomes a restricted version of the algorithm chosen to execute Step 2.

The BOXSTEP method is an acceleration device which should encourage the exploitation of problem structure. For example, successive executions of Step 2 involve problems which are highly related and information from one solution can be used in the next iteration. The well-known advantages of reoptimization should be exploited if possible. These advantages and other important computational considerations are discussed in the context of the more specific applications contained in the subsequent sections.

3. Implementation: solving the local problem by outer approximation.

We now specify that for the remainder of this paper Step 2 of the BOXSTEP method is to be executed with an outer approximation (cutting plane) algorithm. Thus, in Geoffrion's framework [ ], each local problem will be solved by Outer Linearization/Relaxation.

Both v and Y can be represented in terms of the family of their linear supports. Thus

$$(3.1) \qquad v(y) = \min_{k \in K} (f^k + g^k y)$$

$$(3.2) \qquad Y = \left\{ y \in R^n \mid p^j + q^j y \geq 0 \quad \text{for } j \in J \right\}$$

where J and K are index sets, p^j and f^k are scalars, and q^j and $g^k \in R^n$. These are such that

a) for each $k \in K$ there is a $y \in Y$ with $v(y) = f^k + g^k y$; and

b) for each $j \in J$ there is a $y \in Y$ with $p^j + q^j y = 0$.

In the applications presented in sections 4, 5, and 6 the function v(y) represents the optimal value of a subproblem that is parameterized on y. The set Y contains all points y for which (SPy) is of interest. When $\hat{y} \in Y$ the algorithm for (SP$\hat{y}$) produces a linear support for v at $\hat{y}$, but if $\hat{y} \notin Y$ it produces a constraint that is violated at $\hat{y}$. Thus in the former case we get f^{k*} and g^{k*} such that

$$(3.3) \qquad v(\hat{y}) = f^{k*} + g^{k*}\hat{y}$$

while in the latter case we get p^{j*} and q^{j*} such that

$$(3.4) \qquad p^{j*} + q^{j*}\hat{y} < 0.$$

Given the representations in (3.1) and (3.2), the local problem at any $y^t \in Y$ can be written as

$P(y^t; \beta)$ $\quad \max_{y,\sigma} \ \sigma$

subject to

$$f^k + g^k y \geq \sigma \qquad \text{for } k \in K$$

$$p^j + q^j y \geq 0 \qquad \text{for } j \in J$$

$$y_i^t - \beta \leq y_i \leq y_i^t + \beta \qquad \text{for } i = 1, \ldots, n .$$

Let $\bar{P}(y^t; \beta)$ denote $P(y^t; \beta)$ with J replaced by a finite $\bar{J} \subset J$ and K replaced by a finite $\bar{K} \subset K$. Thus $\bar{P}(y^t; \beta)$ is a linear program and a relaxation of $P(y^t; \beta)$. The outer approximation algorithm for Step 2 of the BOXSTEP method can then be stated as follows.

Step 2a: Choose an initial $\bar{J}$ and $\bar{K}$.

Step 2b: Solve the linear program $\bar{P}(y^t; \beta)$ to obtain an optimal solution $(\hat{y}, \hat{\sigma})$.

Step 2c: If $\hat{y} \in Y$, continue to 2d. Otherwise determine p^{j*} and q^{j*} such that

$$p^{j*} + q^{j*}\hat{y} < 0.$$

Set $\bar{J} = \bar{J} \cup \{j*\}$ and go to 2b.

Step 2d: Determine f^{k*} and g^{k*} such that

$$v(\hat{y}) = f^{k*} + g^{k*}\hat{y} .$$

If $v(\hat{y}) < \hat{\sigma} - \epsilon$, set $\bar{K} = \bar{K} \cup \{k*\}$ and go to 2b.

Step 2e: Done; $(\hat{y}, \hat{\sigma})$ is ϵ-optimal for $P(y^t; \beta)$.

The convergence properties of this procedure are discussed in Zangwill [ ] and in Luenberger [ ].

It is not necessary to solve every $P(y^t; \beta)$ to completion (i.e., ϵ-optimality). If a tolerance $\Delta > 0$ is given and if at Step 2d we find that

(3.5) $\quad v(\hat{y}) > v(y^t - \Delta$

then we could immediately move the center of the box to $\hat{y}$ and set $y^{t+1} = \hat{y}$. This would eliminate unnecessary work far from the global maximum.

Step 2b requires the solution of a linear program. The role of reoptimization within a single execution of Step 2 is obvious. Between successive executions of Step 2 there is an opportunity for reoptimization arising from the fact that the constraints indexed by J and K are valid globally. Thus at Step 2a we may choose the initial $\bar{J}$ and $\bar{K}$ to include any constraints generated in earlier boxes. An obvious choice is to retain any constraint that was binding in the final optimal tableau of the preceding local problem.

The question of whether or not to carry over constraints from one local problem to the next is, in fact, an empirical one and the answer is highly dependent on problem structure. For the application discussed in section 5 it was clearly best to save old constraints and omit the line search between boxes. This was because of the substantial computational burden involved in reoptimizing the subproblem (SPy). In the application of section 4, however, it proved best to discard all constraints upon completing a local problem and to use the line search before placing the next box.

In the case where v(y) is the optimal value of (SPy), there is also an opportunity to use reoptimization techniques on (SPy). When this is possible the BOXSTEP method is especially attractive. This is because the successive points y for which the solution of (SPy) is needed are prevented, by the box, from being very far apart. Suppose, for example, that (SPy) is an out-of-kilter problem whose arc costs depend on y. Reoptimization is then very fast for small changes in y but may equal the solution-from-scratch time for large changes in y.

Finally, the desire to use the framework of column generation for the reoptimization techniques at Step 2b dictates that we work on the dual of $\bar{P}(y^t; \beta)$. We record this dual here for future reference.

$$(3.6) \qquad \min \sum_{k \in \bar{K}} f^k \lambda_k + \sum_{j \in \bar{J}} p^j \mu_j - \sum_{i=1}^{n} \left(y_i^t - \beta \right) \delta_i^+ + \sum_{i=1}^{n} \left(y_i^t + \beta \right) \delta_i^-$$

$$\text{s.t.} \sum_{k \in \bar{K}} \lambda_k = 1$$

$$\sum_{k \in \bar{K}} (-g^k) \lambda_k + \sum_{j \in \bar{J}} (-q^j) \mu_j - I\delta^+ + I\delta^- = 0$$

$$\lambda, \mu, \delta^+, \delta^- \geq 0 .$$

The similarity of (3.6) to a Dantzig-Wolfe master problem will be commented upon in the next section.

All of the computational results presented in the subsequent sections were obtained by implementing the BOXSTEP method, as described above, within the SEXOP linear programming system [ ].

4. Application: price-directive decomposition

Consider the linear program

$$(DW) \qquad \min_{x \in X} \; cx \quad \text{s.t.} \quad Ax \leq b$$

where X is a non-empty polytope and A is an (mxn) matrix. This is the problem addressed by the classical Dantzig-Wolfe decomposition method [ ]. It is assumed that the constraints determined by A are coupling or "complicating" constraints in the sense that it is much easier to solve the Lagrangian subproblem

$$(SPy) \qquad \min_{x \in X} \; cx + y(Ax - b)$$

for a given value of y than to solve (DW) itself. If we let v(y) denote the minimal value of (SPy), then the dual of (DW) with respect to the coupling constraints can be written as

$$(4.1) \qquad \max_{y \geq 0} \; v(y).$$

Let $\{x^k \mid k \in K\}$ be the extreme points and $\{z^j \mid j \in J\}$ be the extreme rays of X. Then $v(y) > -\infty$ if and only if $y \in Y'$ where

$$(4.2) \qquad Y' \equiv \{y \in R^m \mid cz^j + yAz^j \geq 0 \quad \text{for } j \in J\}$$

and when $y \in Y'$ we have

$$(4.3) \qquad v(y) = \min_{k \in K} \; (cx^k + yAx^k) - yb$$

The set Y of interest is the intersection of Y' with the non-negative orthant, since (4.1) specifies that y be non-negative. Thus Y and v(y) are of the form discussed in section 3, except for the yb term in (4.3). (Note also that y is a row vector.) In this context the local problem is

(4.4) $\max \ \sigma - yb$

s.t.

$$\sigma - y(Ax^k) \le cx^k \quad \text{for } k \in K$$

$$- y(Az^j) \le cz^j \quad \text{for } j \in J$$

$$-y_i \le -\max\left\{0,\ y_i^t - \beta\right\} \quad \text{for } i = 1,\ldots, m$$

$$y_i \le y_i^t + \beta \quad \text{for } i = 1,\ldots, m$$

and the problem solved at Step 2b (see (3.6)) is

$$\text{(4.5)} \quad \min \sum_{k\in K}(cx^k)\lambda_k + \sum_{j\in J}(cz^j)\mu_j - \sum_{i=1}^{m} \max\left\{0,\ y_i^t - \beta\right\}\delta_i^+ + \sum_{i=1}^{m}\left(y_i^t + \beta\right)\delta_i^-$$

s.t.

$$\sum_{k\in K}\lambda_k = 1$$

$$\sum_{k\in K}(-Ax^k)\lambda_k + \sum_{j\in J}(-Az^j)\mu_j - I\delta^+ + I\delta^- = -b$$

$$\lambda,\ \mu,\ \delta^+,\ \delta^- \ge 0 \ .$$

If the point y^t is in fact the origin ($y^t = 0$) then the objective function of (4.5) becomes

$$\text{(4.6)} \quad \sum_{k\in K}(cx^k)\lambda_k + \sum_{j\in J}(cz^j)\mu_j + \sum_{i=1}^{m}\beta\delta_i^-$$

and hence (4.5) becomes exactly the Dantzig-Wolfe master problem. There is a slack variable δ_i^+ and a surplus variable δ_i^- for each row. Since the constraints were $Ax \le b$ each slack variable has zero cost while each surplus variable is assigned the positive cost β. The cost β must be large enough to drive all of the surplus variables out of the solution. In terms of the BOXSTEP method, then, Dantzig-Wolfe decomposition is simply the solution of one local problem over a sufficiently large box centered at the origin. The appearance of a surplus variable in the final solution would

indicate that the cost β was not large enough, i.e., that the box was not large enough.

The test problem that we have used is a linear program of the form (DW) which represents a network design model. The matrix A has one row for each arc in the network. The set X has no extreme rays, hence Y is just the non-negative orthant. The subproblem (SPy) separates into two parts. The first part involves finding all shortest routes through the network. The second part can be reduced to a continuous knapsack problem. For a network with M nodes and L arcs problem (DW), written as a single linear program, has $M(M-1) + 3L + 1$ constraints and $2LM + 3L$ variables. The details of this model are given by Agarwal [ ].

For this type of problem the best performance was obtained by solving each local problem from scratch. Thus constraints from previous local problems were not saved. A line search, as indicated in section 2, was performed between successive boxes. This was done with an adaptation of Fisher and Shapiro's efficient method for concave piecewise linear functions [ ].

Table 1 summarizes our results for a test problem with M = 12 nodes and L = 18 arcs. The problem was run with several different box sizes. Each run started at the same point y^1 - a heuristically determined solution arising from the interpretation of the problem. For each box size β the column headed $\bar{N}(\beta)$ gives the average number of constraints generated per box. Notice that this number increases monotonically as the box size increases. For a fixed box size, the number of constraints generated per box did not appear to increase systematically as we approached the global optimum. The column headed T gives the total computation time, in seconds, for a CDC6400.

Table 1. Solution of network design test problem by BOXSTEP (price directive) with varying box sizes.

β (box size)	no. of boxes required	$\bar{N}(\beta)$	T (seconds)
0.1	34	12.7	172
0.5	18	14.2	118
1.0	13	17.1	104
2.0	9	17.7	88
3.0	6	25.0	99
4.0	4	26.8	76
5.0	5	33.4	134
6.0	4	34.3	115
7.0	3	38.0	119
20.0	2	67.5	203
25.0	2	74.0	243
30.0	1	74.0	128
1000.0	1	97.0	217

The largest box (β = 1000) represents the Dantzig-Wolfe end of the scale. The smallest box (β = 0.1) produces an ascent that is close to being a steepest ascent. A pure steepest ascent algorithm, as proposed by Grinold [ ], was tried on this problem. With Grinold's primal/dual step-size rule the steps became very short very quickly. By taking optimal size steps instead, we were able to climb higher but appeared to be converging to the value 5097. The maximum was at 5665. The poor performance of steepest ascent is consistent with our poor results for very small boxes.

5. Application: resource-directive decomposition

The dual or Lagrangian orientation of the previous section is complemented by the formulation discussed in this section. Here the function v(y) is obtained from the application of Geoffrion's primal resource-directive strategy [ ] to a large structured linear program.

A large-scale contract selection and distribution problem is formulated as a structured mixed integer linear programming problem by Austin and Hogan [ ]. The linear program consists of a large single commodity network with a few resource constraints on some arcs. The integer portion of the problem models a binary decision regarding the inclusion or exclusion of certain arcs in the network. Embedded in a branch-and-bound scheme, the bounding problem is always a network problem with resource constraints. Hogan [ ] has applied a primal resource directive strategy to such problems, resulting in a subproblem of the form

$$(SPy) \qquad \min \sum_r c_r x_r$$

s.t.

$$\sum_{r \in B_i} x_r - \sum_{r \in A_i} x_r = 0 \qquad \text{for all } i$$

$$\ell_r \le x_r \le h_r \qquad \text{for } r \notin R$$

$$\ell_r \le x_r \le \min \{h_r, y_r\} \qquad \text{for } r \in R$$

where

x_r : is the flow on arc r

c_r : is the cost per unit of flow on arc r

ℓ_r : is the lower bound of flow on arc r

h_r : is an upper bound of flow on arc r

B_i : the arcs with node i as sink

A_i : the arcs with node i as source

R : the arcs subject to resource constraints.

For any vector y, v(y) is the minimal value of (SPy). Note that there is one variable y_r for each arc that is resource-constrained. Let

$$(5.1) \qquad Y^1 = \{y \mid (SPy) \text{ is feasible}\},$$

and

$$(5.2) \qquad Y^2 = \{y \mid \sum_{r \in R} a_{ir} y_r \leq b_i \quad \text{for } i = 1, \ldots, m\}.$$

Thus Y^2 is the feasible region (in y-space) determined by the resource constraints. The set Y of interest is then $Y^1 \cap Y^2$.

The piecewise linear convex function v can be evaluated at any point y by solving a single commodity network problem. As a by-product we obtain a linear support for v at y. Similarly, the implicit constraints in Y^1 can be represented as a finite collection of linear inequalities, one of which is easily generated whenever (SPy) is not feasible. Thus v(y) lends itself naturally to an outer approximation solution strategy. The details are given by Hogan [ ].

An outer approximation algorithm was, in fact, implemented for the problem

$$(5.3) \qquad \min_{y \in Y} \; v(y).$$

This algorithm was subsequently embedded in the BOXSTEP method with substantial computational improvement. To examine the effect of varying the box size, twenty five test problems of the type found in [ ] were randomly generated and solved. The basic networks had approximately 650 arcs of which four important arcs were constrained by two resource constraints. In each case, the problem was initialized with a solution of

(SPy) without the resource constraints (i.e., $y_r \geq h_r$) and a randomly generated initial value of y. The mean B6700 seconds to solution are recorded in Table 2 under the column headed T_1. Sporadically available results from larger test problems indicated greater sensitivity of solution times to box sizes.

The theoretic appeal of BOXSTEP in this case arises from three major points which can be classified as: cut requirements, reoptimization of the subproblem, and reoptimization of the local problem.

Cut requirements: The v function is piecewise linear or polyhedral. Hence its epigraph, epi v, has a finite number of faces. To characterize v over any subset of Y, the outer approximation method generates a series of cuts or linear supports. Each cut is coincident with at least one face of epi v and no cut is repeated. The smaller the subset of Y the smaller the number of faces and, therefore, the smaller the number of cuts needed to describe v. It follows that the smaller the box, the smaller the computational burden in solving the local problem at Step 2. This has already been demonstrated by the results of section 4.

Reoptimization of the subproblem: Once an initial solution for the basic network has been obtained, v(y) can be determined by reoptimizing the basic network with changes in some of the arc capacities (i.e., for $r \in R$). If these changes are small, as they must be when β is small, these reoptimizations can be performed quickly. Since most of the computational burden in this problem is devoted to reoptimizing the network, this proves to be a significant consideration.

Reoptimization of the local problem: Since the generation of cuts requires the relatively expensive reoptimization of the subproblem, some or all of

Table 2. Solution of 25 resource constrained network problems by BOXSTEP (resource directive) with varying box sizes

β (box size)	T_1^* (seconds)	T_2^* (seconds)
10^1	31.8	4.1
10^2	23.1	5.8
10^3	21.2	14.9
10^4	34.6	25.0

* T_1 : Mean time to solution using a randomly generated vector as the initial y.

T_2 : Mean time to solution using an optimal solution as the initial y.

these cuts should be retained as the box moves. This greatly reduces the number of cuts that must be generated to solve each local problem after the first one. In contrast to the application presented in section 4, saving cuts paid off but the line search did not. The mechanism for saving cuts was very simple. Up to a fixed number were accumulated. Once that number was reached every new cut replaced an old non-basic cut.

Although the size of the box is inversely related to the effort required to solve the problem within the box, the results indicate a trade-off between the size of the box and the number of moves required to solve the overall problem (5.3). There is a notable exception to this rule however. Frequently, if not always, there is a readily available prior estimate of an optimal solution point y^*. Most large-scale problems have a natural physical or economic interpretation which will yield a reasonable estimate of y^*. In this application, recall that (5.3) is actually the bounding problem in a branch-and-bound scheme. The v function changes slightly as we move from one branch to another. The minimizing point y^* changes little if at all. Thus we wish to solve a sequence of highly related problems of the form (5.3). Using the previous y^* as the starting point on a new branch would seem quite reasonable. Furthermore, the size of the box used should be inversely related to our confidence in this estimate. To illustrate this important point, the twenty five test problems were restarted with the box centered at the optimal solution. The mean time required, in B6700 seconds, to solve the problem over this box is recorded in Table 2 under the column headed T_2. These data and experience with this class of problems in the branch and bound procedure indicate that starting with a good estimate of the solution and a small boxsize reduces

the time required to solve (5.3) by an order of magnitude as compared to the case with $\beta = +\infty$. Clearly a major advantage of the BOXSTEP method is the ability to capitalize on prior information.

Ascent methods typically exploit prior information in the form of a good initial estimate but the information generated during the solution procedure is not cumulative. Outer approximation methods, in contrast, do not exploit prior information but the cuts generated during solution are cumulative. The current applications of the BOXSTEP method fall in the conceptual continuum between these two extremes and capture the best features of both.

6. Application: price-direction revisited

Some of the resource-constrained network problems introduced in section 5 have also been solved by price-directive decomposition as developed in section 4. These results will be presented very briefly. Using the notation of section 5 the problem as given is

$$\text{(6.1)} \qquad \min \sum_{r} c_r x_r$$

$$\text{(6.2)} \qquad \text{s.t.} \sum_{r \in B_i} x_r - \sum_{r \in A_i} x_r = 0 \qquad \text{for all } i$$

$$\text{(6.3)} \qquad \ell_r \leq x_r \leq h_r \qquad \text{for all } r$$

$$\text{(6.4)} \qquad \sum_{r \in R} a_{ir} x_r \leq b_i \qquad \text{for } i = 1, \ldots, m \, .$$

If we dualize with respect to the resource constraints (6.4), the resulting subproblem is a network problem parameterized in its arc costs rather than in its capacities.

Tests were made with a network of 142 nodes and 1551 arcs. The unconstrained solution (which took 5 seconds on the CDC6400) had positive flow on only 100 of the arcs. These 100 arcs were divided into 5 sets of 20 each and 5 resource constraints were constructed. For the price-directive approach, then, y was a 5 dimensional vector. As in section 5 the best performance was obtained by saving old cuts and omitting the line search.

Table 3 contains the results for a series of runs, each starting at the origin. The maximum number of cuts retained was 20. The time is given in CDC6400 seconds. (For this type of application the CDC6400 is between 5 and 6 times faster than the B6700.) In addition, the twenty five problems reported in section 5 were solved by price-directive decomposition on a B6700 and the counterpart of Table 2 is reported in Table 4.

Table 3 . Price-directive results for a resource-constrained network problem.

BOX SIZE (β)	Number of boxes required	T (seconds)
.1	?	> 150
.5	7	70
1.0	4	55
2.0	2	76
3.0	2	119
4.0	1	92
5.0	1	108
1000.0	1	> 150

Table 4. Solution of 25 resource constrained network problems by BOXSTEP (price directive) with varying box sizes.

$\hat{\varepsilon}$ (box size)	T_1^* (seconds)	T_2^* (seconds)
10^0	67.8	3.7
10^1	35.1	4.8
10^2	17.0	9.9
10^3	22.4	14.9
10^4	27.1	17.0

* T_1 : Mean time to solution using a randomly generated vector as the initial y.

T_2 : Mean time to solution using an optimal solution as the initial y.

7. Conclusion

We have only scratched the surface as far as applications of the BOXSTEP method are concerned. The main avenues for future work appear to be the following.

Structured linear programs: Many other problems for which decomposition has failed in the past need to be re-investigated. This is especially true when BOXSTEP is considered in conjunction with other new developments in large-scale optimization (more on this below).

Structured non-linear programs: BOXSTEP has yet to be tried on non-linear problems. Two very likely starting points are Geoffrion's tangential approximation approach to the resource-directive decomposition of non-linear systems [ ] and Geoffrion's generalized Benders decomposition method [ ].

General non-linear programs: In the case where v(y) is an explicitly available concave function, BOXSTEP could perhaps be used to accelerate the convergence of any outer approximation or cutting plane algorithm. This has not yet been tried. There may be other kinds of algorithms that can profitably be embedded in the BOXSTEP method.

Integer programming: Geoffrion [ ] and Fisher and Shapiro [ ] have recently shown how the maximization of Lagrangian functions can provide strong bounds in a branch-and-bound framework. The BOXSTEP method should find many fruitful applications in this context. It has the desirable property that the maximum values for successive boxes form a monotonically increasing sequence.

There are also several tactical questions to be investigated. Is there a rule-of-thumb for the best box size to use for a given problem? When should cuts be saved? When is the line search beneficial? We shall

confine ourselves here to one such question: How should the starting point y^1 be chosen?

The BOXSTEP method can take advantage of a good starting point. This point may be derived heuristically from the interpretation of the problem, as in section 4, or it may be taken as the optimal solution of a closely related problem, as in section 5. Alternatively, we can start BOXSTEP where some other algorithm leaves off. Suppose that an algorithm with good initial behavior but slow convergence is applied to the given problem. As soon as this algorithm stops making satisfactory progress a switch can be made to the BOXSTEP method. This has been tried with dramatic success. BOXSTEP has been coupled with the subgradient relaxation method recently introduced by Held and Karp [ ] and Held and Wolfe [ ]. A sample of the results will indicate the benefits that may be derived from this kind of hybrid algorithm.

Recall the representation (3.1) of $v(y)$ in terms of its linear supports. If $v(y^t) = f^{k(t)} + g^{k(t)}y^t$, then it is well-known that $g^{k(t)}$ is a subgradient of v at y^t. Held and Wolfe propose the following iterative process starting at any point y^1:

$$y^{t+1} = y^t + s_t g^{k(t)} \tag{7.1}$$

where $g^{k(t)}$ is a subgradient of v at y^t and $\{s_t\}_{t=1}^{\infty}$ is a sequence for which $s_t \to 0$ but $\sum_{t=1}^{\infty} s_t = \infty$. Any point generated by this process could be taken as the starting point for BOXSTEP. The hybrid algorithm was tested on the p-median problem (see Marsten [ ] or ReVelle and Swain [ ]). The continuous version of this problem has the form

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{7.2}$$

s.t.

$$(7.3) \qquad \sum_{j=1}^{n} x_{jj} = p$$

$$(7.4) \qquad x_{ij} \le x_{jj} \qquad \text{for all } i \ne j$$

$$(7.5) \qquad 0 \le x_{ij} \le 1 \qquad \text{for all } i, j$$

$$(7.6) \qquad \sum_{j=1}^{n} x_{ij} = 1 \qquad \text{for } i = 1, \ldots, n .$$

The price-directive approach developed in section 4 was applied to this problem. Dualizing with respect to the constraints (7.6) results in a Lagrangian subproblem for which a very efficient solution recovery technique has been devised by Blankenship [ ]. The starting point for BOXSTEP was obtained by making 250 steps of the process (7.1). The sequence $\{s_t\}$ was taken as five repetitions of 40/t for each t = 1,..., 50. The results are given in Table 5 . The column headed p gives the median sought, as specified in (7.3). The columns headed L_{125} and L_{250} give the value of the Lagrangian v(y) after 125 and after 250 steps. L_{max} is the maximum value of the Lagrangian. T_1 and T_2 give the time devoted to (7.1) and to BOXSTEP respectively in CDC6400 seconds. The last column gives the number of boxes used by the BOXSTEP method. This test problem had n = 33 and the box size $\beta = 1$ was used in each case. (For p = 2 and 4 an optimal solution and zero subgradient were obtained very quickly, making the subsequent steps and BOXSTEP solution superfluous.) These results indicate that by using more than strictly local information BOXSTEP is able to attain and verify the optimal solution quickly. Hybrid algorithms of this type should produce significant computational breakthroughs in the very near future.

Table 5 . Results for p-median problem with hybrid algorithm

p	L_{125}	L_{250}	L_{max}	T_1	T_2	boxes
2	17474.0	17474.0	17474.0	9.2	0.3	1
3	14538.7	14622.5	14627.0	8.9	19.6	3
4	12363.0	12363.0	12363.0	11.8	0.2	1
8	7449.1	7454.0	7460.0	13.6	8.4	3
9	6840.9	6843.6	6846.0	11.5	3.3	2
10	6263.8	6265.4	6267.0	14.9	0.7	2
11	5786.8	5786.9	5787.0	15.2	1.0	2
20	2779.7	2785.9	2786.0	14.5	0.9	2
30	509.8	514.3	515.0	14.6	1.7	2

References

[ ] Agarwal, S. K., (1973). "Optimizing Techniques for the Interactive Design of Transportation Networks Under Multiple Objectives", Ph.D. dissertation, Dept. of Civil Engineering, Northwestern University.

[ ] Austin, L. M. and W. W. Hogan, (1973). "Optimizing Procurement of Aviation Fuels for the Defense Supply Agency", Working Paper, U. S. Air Force Academy, June.

[ ] Blankenship, J. W. (1973). "BOXSTEP: A New Approach to La Grangean Decomposition", Ph.D. dissertation, Department of Industrial Engineering and Management Sciences, Northwestern University.

[ ] Beale, E. M. (1971). "A Conjugate Gradient Method of Approximation Programming", Scientific Control Systems Limited, London.

[ ] Dantzig, G. B. and P. Wolfe, (1960). "Decomposition Principles for Linear Programs", Operations Research, Vol. 8, No. 1 (Jan.-Feb.), pp. 101-111.

[ ] Fisher, M. L. and J. F. Shapiro, (1973). "Constructive Duality for Discrete Optimization", Unpublished paper, Graduate School of Business, University of Chicago, (March).

[ ] Geoffrion, A. M. (1970). "Elements of Large-Scale Mathematical Programming", Management Science, Vol. 16, No. 11, (July), pp. 652-691.

[ ] Geoffrion, A. M., (1970). "Primal Resource-Directive Approaches for Optimizing Nonlinear Decomposable Systems", Operations Research, Vol. 18, pp. 375-403.

[ ] Geoffrion, A. M., (1970). "Generalized Benders Decomposition", Working Paper No. 159, Western Management Science Institute, U.C.L.A.

[ ] Griffith, R. E. and R. A. Stewart (1971). "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems", Management Science, 7, pp. 379-392.

[ ] Grinold, R. C., (1972). "Steepest Ascent for Large Scale Linear Programs", SIAM Review, Vol. 14, No. 3 (July), pp. 447-464.

[ ] Held, M. and R. M. Karp, (1971). "The Traveling-Salesman Problem and Minimum Spanning Trees: Part II", Mathematical Programming, 1, pp. 6-25.

[ ] Held, M. and P. Wolfe, (1973). "Large-Scale Optimization and the Relaxation Method", IBM Research, Yorktown Heights, New York.

[ ] Hogan, W. W., (1973). "Single Commodity Network Problems With Resource Constraints", Discussion Paper, Department of Economics and Management, United States Air Force Academy.