LIBRARY
OF THE
MASSACHUSETTS INSTITUTE
OF TECHNOLOGY
WORKING PAPER
ALFRED P. SLOAN SCHOOL OF MANAGEMENT

The BOXSTEP Method for
Large Scale Optimization

660-73

June 1973

(Revised December 1973)

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

MASSACHUSETTS
INSTITUTE OF TECHNOLOGY
50 MEMORIAL DRIVE
CAMBRIDGE, MASSACHUSETTS 02139
The BOXSTEP Method for Large Scale Optimization

660-73

June 1973

(Revised December 1973)

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

*Federal Energy Office

**Sloan School of Management, M.I.T.

***Institute for Defense Analyses
Contents

1. Introduction. .................................... 1
2. The BOXSTEP Method. ............................ 3
3. Implementation: solving the local problem by outer approximation. ............................ 7
4. Application: Dantzig-Wolfe decomposition ...... 11
5. Application: Benders decomposition ........... 16
7. Application: Subgradient optimization .......... 27
8. Conclusion. .................................... 31
1. Introduction

This paper presents a simple heuristic that has unified some previously separate areas of theory and has produced some surprising computational advantages. The underlying 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 \( (SP_y) \) that is parameterized on \( y \). Thus \( v(y) \) is not explicitly available and evaluating it at \( y \) means solving \( (SP_y) \). In this context, the motivation for the method lies in the empirical observation that the number of times \( (SP_y) \) must be solved in order to maximize \( v(y) \) over \( Y \cap B \) can be controlled by adjusting the size of the box \( B \). This behavior is extremely important in decomposition methods or other problems where the evaluation of \( v(y) \) imposes a real computational burden.
We begin by presenting BOXSTEP 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 [11] and recently revived by Beale [4] and Meyer [20]. Section 2 presents the BOXSTEP method in very general terms and proves its convergence. Section 3 shows how an outer approximation 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 through 7 contain specific applications to large structured linear programs. Section 8 points out some 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 $\epsilon$-optimal termination, and discuss a modification of the basic method which will not upset the fundamental convergence property.

Consider any problem of the form

$$(P) \quad \max_{y \in Y} v(y), \text{ with } Y \subseteq \mathbb{R}^n \text{ and } v:Y \rightarrow \mathbb{R}. $$

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

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

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 $\epsilon$-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 $2\beta$ 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 have multiple maxima, 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. Assuming $Y$ to be compact, let

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

(the "diameter" of $Y$),

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

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 $2\varepsilon/\lambda$-optimal solution.

Proof: First we establish that the method terminates finitely. If $\varepsilon > 0$, then non-termination implies that $v(y^{t+1}) \geq v(y^1) + t\varepsilon$ 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 $\varepsilon = 0$, 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 $||y^s - y^t||_\infty > \beta/2$ 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,

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

Now the definition of $\lambda$ implies that

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

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

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

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.

If $v(y)$ is piecewise linear, then we may take $\epsilon = 0$. In general, however, the requirement that we obtain an $\epsilon$-optimal solution of each local problem means that $\epsilon$ must be strictly positive.
As we shall see below, a number of alternatives are available for determining the next position of the box after each local optimization. Moving the center of the box from \( y^t \) to \( y^{t+1} \) is the simplest possible tactic. An important alternative would be to perform a line search in the direction \( d^t = y^{t+1} - y^t \). At Step 3 we would replace \( y^{t+1} \) by \( y^t + \theta^t d^t \) where \( \theta^t \) is optimal for

\[
(\text{LS}) \quad \max_{\theta \geq 1} v(y^t + \theta d^t) \quad \text{s.t.} \quad y^t + \theta d^t \in Y.
\]

This modification of the basic method does not require any change in the statement of the theorem. Far from the global optimum, therefore, BOXSTEP could be viewed as a feasible directions method which uses more than strictly local information to determine the next direction of search. Once the global optimum is contained in the current box, however, BOXSTEP is simply a restricted version of the algorithm chosen to execute Step 2.
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 the framework of [7], each local problem will be solved by Outer Linearization/Relaxation.

By convexity, both v and Y can be represented in terms of the family of their linear supports. Thus

\[ v(y) = \min_{k \in K} (f^k + g^k y) \tag{3.1} \]
\[ Y = \{y \in \mathbb{R}^n | p^j + q^j y \geq 0 \text{ for } j \in J \} \tag{3.2} \]

where J and K are index sets, p^j and f^k are scalars, and q^j, g^k \in \mathbb{R}^n.

These are such that

a) for each y \in Y there is a k \in K with v(y) = f^k + g^k y; and
b) for each y on the boundary of Y there is a j \in J with p^j + q^j y = 0.

In the applications presented in sections 4 through 7 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

\[ v(\hat{y}) = f^{k*} + g^{k*} \hat{y} \tag{3.3} \]

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

\[ p^{j*} + q^{j*} y < 0. \tag{3.4} \]

Given the representations in (3.1) and (3.2), the local problem at any \( y^t \in Y \) can be written as
\[ P(y^t; \beta) \max_{y, \sigma} \]

subject to
\[ f^k + g^k y \geq \sigma \quad \text{for } k \in K \]
\[ p^j + q^j y \geq 0 \quad \text{for } j \in J \]
\[ y^t_i - \beta \leq y_i \leq y^t_i + \beta \quad \text{for } i = 1, \ldots, n. \]

Let \( \overline{P}(y^t; \beta) \) denote \( P(y^t; \beta) \) with \( J \) replaced by a finite \( \overline{J} \subset J \) and \( K \) replaced by a finite \( \overline{K} \subset K \). Thus \( \overline{P}(y^t; \beta) \) 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 \( \overline{J} \) and \( \overline{K} \).

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

**Step 2c:** If \( \overline{y} \notin Y \), continue to 2d. Otherwise determine \( p^{j^*} \) and \( q^{j^*} \) such that
\[ p^{j^*} + q^{j^*} y < 0. \]
Set \( \overline{J} = \overline{J} \cup \{j^*\} \) and go to 2b.

**Step 2d:** Determine \( f^{k^*} \) and \( g^{k^*} \) such that
\[ v(\overline{y}) = f^{k^*} + g^{k^*} y. \]
If \( v(\overline{y}) < \overline{\sigma} - \varepsilon \), set \( \overline{K} = \overline{K} \cup \{k^*\} \) and go to 2b.

**Step 2e:** Done; \((\overline{y}, \overline{\sigma})\) is \( \varepsilon \)-optimal for \( P(y^t; \beta) \).

The convergence properties of this procedure are discussed, for example, in Zangwill [22] or Luenberger [17].

It is not necessary to solve every \( P(y^t; \beta) \) to completion (i.e., \( \varepsilon \)-optimality). If a tolerance \( \Delta > 0 \) is given and if at Step 2d we find that
\[ v(\overline{y}) > v(y^t) + \Delta \]

(3.5)
then we could immediately move the center of the box to \( \hat{y} \) and set \( y^{t+1} = \hat{y} \). This is one of many heuristics that would eliminate unnecessary computation 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. However, between successive executions of Step 2, i.e. between successive local problems, 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 and accumulate a description of \( v \) and \( Y \).

In the case where \( v(y) \) is the optimal value of (SPy), the power of the BOXSTEP method rests upon two observations. First, the number of times that (SPy) must be solved in order to solve a local problem can be controlled by adjusting the size of the box. For a larger box, more solutions of (SPy) (i.e. linear supports of \( v \) and \( Y \)) are required. This empirical finding is documented in the computational results which follow. Second, reoptimization of (SPy) is greatly facilitated because the successive points \( y \) for which the solution of (SPy) is needed are prevented, by the box, from being very far apart. For example, when (SPy) is a network problem whose arc costs depend on \( y \) reoptimization is very fast for small changes in \( y \) even though it may equal the solution-from-scratch time for large changes in \( y \).

Ascent and feasible directions 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 movement of the box and the accumulation of cuts place the BOXSTEP method in the conceptual continuum between these two extremes and captures features of both.
The desire to use the framework of column generation for the reoptimization techniques at Step 2b dictates that we work on the dual of $P(y^t; \beta)$. We record this dual here for future reference.

\[
(3.6) \quad \min \sum_{k \in K} f^k \lambda_k + \sum_{j \in J} p^j \mu_j - \sum_{i=1}^{n} (y^t_1 - \beta) \delta^+_i + \sum_{i=1}^{n} (y^t_1 + \beta) \delta^-_i \\
\text{s.t.} \quad \sum_{k \in K} \lambda_k = 1 \\
\sum_{k \in K} (-g^k) \lambda_k + \sum_{j \in J} (-q^j) \mu_j - I \delta^+ + I \delta^- = 0
\]

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

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 [19].
4. Application: Dantzig-Wolfe decomposition

Consider the linear program

\[(DW) \quad \min_{x \in X} cx \quad s.t. \quad Ax \leq b\]

where \(X\) is a non-empty polytope and \(A\) is an \((m \times n)\) matrix. This is the problem addressed by the classical Dantzig-Wolfe decomposition method [5]. 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 Lagrangean subproblem

\[(SPy) \quad \min_{x \in X} cx + y(Ax - b)\]

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

\[(4.1) \quad \max_{y \geq 0} v(y).\]

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

\[(4.2) \quad Y' = \{y \in \mathbb{R}^m | cz^j + yAz^j \geq 0 \quad \text{for} \quad j \in J\}\]

and when \(y \in Y'\) we have

\[(4.3) \quad v(y) = \min_{k \in K} cx^k + y(Ax^k - b)\]

The set \(Y\) of interest is the intersection of \(Y'\) with the non-negative orthant. Thus \(Y\) and \(v(y)\) are of the form discussed in section 3. (Note that \(y\) is a row vector.) In this context the local problem is to maximize
the Lagrangean function $v(y)$ over a box centered at $y^t$. This can be expressed as:

\[
(4.4) \quad \max \sigma - yb \\
\text{s.t.} \\
\sigma - y(Ax^k) \leq cx^k \quad \text{for } k \in K \\
-y(Az^j) \leq cz^j \quad \text{for } j \in J \\
\max \{0, y_{i-1}^t - y^i \leq y_{i+1}^t + \beta \} \quad \text{for } i = 1, \ldots, m
\]

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

\[
(4.5) \quad \min \sum_{k \in K} (cx^k)\lambda_k + \sum_{j \in J} (cz^j)\mu_j - \sum_{i = 1}^m \max \{0, y_{i-1}^t - y^i \} \delta_i^+ + \sum_{i = 1}^m (y_{i+1}^t + \beta)\delta_i^-
\text{s.t.} \\
\sum_{k \in K} \lambda_k = 1 \\
\sum_{k \in K} (-Ax^k)\lambda_k + \sum_{j \in J} (-Az^j)\mu_j - I\delta^+ + I\delta^- = -b \\
\lambda, \mu, \delta^+, \delta^- \geq 0.
\]

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

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

and hence, if $\beta$ is very large, (4.5) becomes exactly the Danzig-Wolfe Master problem. There is a slack variable $\delta_i^+$ and a surplus variable $\delta_i^-$ for each row. Since the constraints were $Ax \leq b$ each slack variable has zero cost while each surplus variable is assigned the positive cost $\beta$. The cost $\beta$ 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 $\beta$ 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. Unfortunately, however, there was no convenient way to reoptimize (SPy). 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 [1].

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 [6].

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 $\beta$ the column headed $\tilde{N}(\beta)$ gives the average number of times (SPy) was solved per box. Notice that this number increases monotonically as the box size increases. For a fixed box size, the number of subproblem solutions per box did not appear to increase systematically as we approached the global optimum. The column headed $T$ gives
Table 1. Solution of network design test problem by BOXSTEP (Dantzig-Wolfe) with varying box sizes.

<table>
<thead>
<tr>
<th>$\beta$ (box size)</th>
<th>no. of boxes required</th>
<th>$\bar{N}(\beta)$</th>
<th>T (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>34</td>
<td>12.7</td>
<td>172</td>
</tr>
<tr>
<td>0.5</td>
<td>18</td>
<td>14.2</td>
<td>118</td>
</tr>
<tr>
<td>1.0</td>
<td>13</td>
<td>17.1</td>
<td>104</td>
</tr>
<tr>
<td>2.0</td>
<td>9</td>
<td>17.7</td>
<td>88</td>
</tr>
<tr>
<td>3.0</td>
<td>6</td>
<td>25.0</td>
<td>99</td>
</tr>
<tr>
<td>4.0</td>
<td>4</td>
<td>26.8</td>
<td>76</td>
</tr>
<tr>
<td>5.0</td>
<td>5</td>
<td>33.4</td>
<td>134</td>
</tr>
<tr>
<td>6.0</td>
<td>4</td>
<td>34.3</td>
<td>115</td>
</tr>
<tr>
<td>7.0</td>
<td>3</td>
<td>38.0</td>
<td>119</td>
</tr>
<tr>
<td>20.0</td>
<td>2</td>
<td>67.5</td>
<td>203</td>
</tr>
<tr>
<td>25.0</td>
<td>2</td>
<td>74.0</td>
<td>243</td>
</tr>
<tr>
<td>30.0</td>
<td>1</td>
<td>74.0</td>
<td>128</td>
</tr>
<tr>
<td>1000.0</td>
<td>1</td>
<td>97.0</td>
<td>217</td>
</tr>
</tbody>
</table>
the total computation time, in seconds, for a CDC6400. All runs were made with $\epsilon=10^{-6}$.

The largest box ($\beta = 1000$) represents the Dantzig-Wolfe end of the scale. The smallest box ($\beta = 0.1$) produces an ascent that approximates a steepest ascent. A pure steepest ascent algorithm, as proposed by Grinold [12], 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 although the maximum was at 5665. The poor performance of steepest ascent on this problem is consistent with our poor results with the smallest box.

Some additional computational results with Dantzig-Wolfe decomposition are reported in sections 6 and 7.
5. Application: Benders decomposition

The dual or Lagrangean orientation of the previous section is complemented by the primal formulation discussed in this section. Here the function \( v(y) \) is obtained from the application of Benders decomposition to a large structured linear program.

A large-scale contract selection and distribution problem is formulated as a mixed integer linear programming problem by Austin and Hogan [2]. The integer portion of the problem models a binary decision regarding the inclusion or exclusion of certain arcs in a network. Embedded in a branch-and-bound scheme, the bounding problem is always a network problem with resource constraints which has the following form.

\[
\begin{align*}
\text{(5.1)} & \quad \min \sum_{k \in K} c_k x_k \\
\text{subject to} & \\
\text{(5.2)} & \quad \sum_{k \in B_i} x_k - \sum_{k \in A_i} x_k = 0 \quad i \in N \\
\text{(5.3)} & \quad l_k \leq x_k \leq h_k \quad k \in K \\
\text{(5.4)} & \quad \sum_{k \in R} a_{jk} x_k \leq r_j \quad j = 1, \ldots, p
\end{align*}
\]

where

- \( N \): the set of nodes
- \( K \): the set of arcs
- \( x_k \): the flow on arc \( k \)
- \( c_k \): the unit cost of flow on arc \( k \)
- \( h_k \): the upper bound for flow on arc \( k \)
- \( l_k \): the lower bound for flow on arc \( k \)
- \( B_i \): the arcs which end at node \( i \)
- \( A_i \): the arcs which originate at node \( i \)
- \( p \): the number of resource constraints
- \( R \): the arcs which are subject to the resource constraints
- \( r_j \): the amount of resource \( j \) available
- \( a_{jk} \): the coefficient of arc \( k \) in resource constraint \( j \).
Hogan [16] has applied Benders decomposition to this problem, resulting in a subproblem of the form

\[(SPy) \quad \min \sum_{k \in K} c_k x_k \]

subject to

\[\sum_{k \in B_i} x_k - \sum_{k \in A_i} x_k = 0 \quad i \in N\]

\[l_k \leq x_k \leq h_k \quad k \in K-R\]

\[l_k \leq x_k \leq \min \{h_k, y_k\} \quad k \in R\]

where \(y_k\) represents an allocation of resources to arc \(k\). For any vector \(y\), \(v(y)\) is the minimal value of \((SPy)\). Note that there is one variable \(y_k\) for each arc that is resource-constrained. Let

\[(5.5) \quad Y^1 = \{y\mid (SPy) \text{ is feasible}\}\]

and

\[(5.6) \quad Y^2 = \{y\mid \sum_{k \in R} a_{jk} y_k \leq r_j \quad j = 1, \ldots, p\}\]

Thus \(Y^2\) is the feasible region (in \(y\)-space) determined by the resource constraints. The set \(Y\) of interest is then \(Y^1 \cap Y^2\), and \((5.1)-(5.4)\) is equivalent to \((5.7)\).

\[(5.7) \quad \min_{y \in Y} v(y)\]

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 [16].
The generation of linear supports or cuts requires the reoptimization of the subproblem. Since this was relatively expensive computationally, some of the cuts were retained as the box moved. This greatly reduced the number of cuts that had to be generated to solve each local problem after the first move of the box. In contrast to the application presented in section 4, saving cuts paid off but the line search did not. The rule for saving cuts was quite simple. Up to a fixed number were accumulated. Once that number was reached each new cut replaced an (arbitrary) old cut that was currently slack.

Twenty-five test problems of the type found in [2] were randomly generated and solved. The networks had approximately 650 arcs of which four important arcs were constrained by two resource constraints. In each case the BOXSTEP method was started at a randomly generated initial value of $y$. The mean B6700 seconds to solution are recorded in Table 2 under the column headed $T_1$. Test runs were made for several box sizes. The results indicate the superiority of a "moderate" sized box, but the computational advantage gained is not as marked as with the real-life problem reported below. All of these runs were made with $\varepsilon=10^{-3}$.

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 (5.7). There is a notable exception to this rule however. Frequently, if not always, a real problem provides 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 the present application, recall that (5.7) is actually the bounding problem in a branch-and-bound scheme. The $v$ function changes slightly as we move from one branch to
another but the minimizing point \( y^* \) changes little if at all. Thus we wish to solve a sequence of highly related problems of the form (5.7). 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 box size reduces the time required to solve (5.7) 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.

A real contract selection and distribution problem was solved for the Defense Supply Agency. This problem had 709 nodes, 4837 arcs, and 6 resource constraints involving 48 of the arcs. There were no integer variables, however, so only one problem of the form (5.7) had to be solved. Two innovations were made in applying the BOXSTEP method to this problem. The first innovation was to use a two-pass solution strategy. The problem was first solved with a relatively large tolerance \( \varepsilon_1 \). Then, starting at the first pass solution, the box size was reduced and the problem resolved with a finer tolerance \( \varepsilon_2 \). This was quite effective, as can be seen in Table 3 where the total solution time in B6700 seconds is given in the column headed \( T \).

The second innovation was to use a non-cubical box. A different size \( \beta_k \) can be selected for each dimension of the box so that we have

\[
(5.5) \quad y_k^o - \beta_k \leq y_k \leq y_k^o + \beta_k \quad \text{for } k \in K
\]
defining a hyper-rectangle centered at $y^0$. In this particular problem each resource-constrained arc appears in only one of the constraints (5.4). For each arc $k$ appearing in constraint $j$ ($a_{jk} \neq 0$), we can take $\beta_k$ as some fraction of the resource availability $r_j$. This was done and the results are reported in Table 4. The best time for this problem was the 1200 seconds obtained by using $r/20$ to set the initial box dimensions.
Table 2. Solution of 25 resource constrained network problems by BOXSTEP (Benders) with varying box sizes.

<table>
<thead>
<tr>
<th>$P$ (box size)</th>
<th>$T_1^*$ (seconds)</th>
<th>$T_2^*$ (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^1$</td>
<td>31.8</td>
<td>4.1</td>
</tr>
<tr>
<td>$10^2$</td>
<td>23.1</td>
<td>5.8</td>
</tr>
<tr>
<td>$10^3$</td>
<td>21.2</td>
<td>14.9</td>
</tr>
<tr>
<td>$10^4$</td>
<td>34.6</td>
<td>25.0</td>
</tr>
</tbody>
</table>

$T_1^*$: 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 $y$. 
Table 3. Solution of DSA problem by BOXSTEP (Benders) with 2-pass strategy.\(^1\)

<table>
<thead>
<tr>
<th>(\varepsilon_1)</th>
<th>(\beta_1)</th>
<th>(\varepsilon_2)</th>
<th>(\beta_2)</th>
<th>(T) (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(10^{-3})</td>
<td>(10^7)</td>
<td>(10^{-5})</td>
<td>(10^7/2)</td>
<td>1450</td>
</tr>
<tr>
<td>(10^{-3})</td>
<td>(10^7)</td>
<td>(10^{-5})</td>
<td>(10^6)</td>
<td>1470</td>
</tr>
<tr>
<td>(10^{-3})</td>
<td>(10^7)</td>
<td>(10^{-5})</td>
<td>(10^7)</td>
<td>1700</td>
</tr>
<tr>
<td>(10^{-3})</td>
<td>(10^{10})</td>
<td></td>
<td></td>
<td>&gt;1800</td>
</tr>
<tr>
<td>(10^{-5})</td>
<td>(10^7)</td>
<td></td>
<td></td>
<td>&gt;1800</td>
</tr>
</tbody>
</table>

\(^1\)No significance should be attached to the large absolute magnitude of \(\beta\). The optimal solution is of the same order of magnitude, \(10^7\).
Table 4. Solution of DSA problem by BOXSTEP (Benders) with 2-pass strategy and rectangular box.

<table>
<thead>
<tr>
<th>$\varepsilon_1$</th>
<th>$\beta_1$</th>
<th>$\varepsilon_2$</th>
<th>$\beta_2$</th>
<th>T (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-3}$</td>
<td>r/5</td>
<td>$10^{-5}$</td>
<td>$8_1/5$</td>
<td>2100</td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>r/10</td>
<td>$10^{-5}$</td>
<td>$8_1/5$</td>
<td>1400</td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>r/20</td>
<td>$10^{-5}$</td>
<td>$8_1/5$</td>
<td>1200</td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>r/50</td>
<td>$10^{-5}$</td>
<td>$8_1/5$</td>
<td>1300</td>
</tr>
<tr>
<td>(Saturated) T</td>
<td>$N^B$</td>
<td>$N^C$</td>
<td>$L^A$</td>
<td>$L^2$</td>
</tr>
<tr>
<td>--------------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>0011</td>
<td>$2 \sqrt{2}$</td>
<td>$2 \sqrt{3}$</td>
<td>$2 \sqrt{7}$</td>
<td>$2 \sqrt{10}$</td>
</tr>
<tr>
<td>0010</td>
<td>$2 \sqrt{2}$</td>
<td>$2 \sqrt{3}$</td>
<td>$2 \sqrt{10}$</td>
<td>$2 \sqrt{10}$</td>
</tr>
<tr>
<td>0011</td>
<td>$2 \sqrt{2}$</td>
<td>$2 \sqrt{3}$</td>
<td>$2 \sqrt{10}$</td>
<td>$2 \sqrt{10}$</td>
</tr>
<tr>
<td>0001</td>
<td>$2 \sqrt{2}$</td>
<td>$2 \sqrt{3}$</td>
<td>$2 \sqrt{10}$</td>
<td>$2 \sqrt{10}$</td>
</tr>
</tbody>
</table>

Some of the resource-constrained network problems introduced in section 5 have also been solved by Dantzig-Wolfe decomposition as developed in section 4. These results will be presented very briefly. If problem (5.1)-(5.4) is dualized with respect to the resource constraints (5.4) the resulting subproblem is a network problem parameterized in its arc costs.

\[(SP_y) \quad \min \sum_{x_k \in K-R} c_k x_k + \sum_{k \in R} \left[ c_k + \sum_{j=1}^{p} y_j a_{jk} \right] x_k - \sum_{j=1}^{p} y_j r_j \]

subject to

\[\sum_{k \in B_i} x_k - \sum_{k \in A_i} x_k = 0 \quad i \in N\]

\[l_k \leq x_k \leq h_k \quad k \in K\]

where \(y\) is a vector of resource prices. Note that there is one variable \(y_j\) for each resource constraint.

Tests were made with a network of 142 nodes and 1551 arcs. Solving this network without resource constraints ((5.1)-(5.3)) took 5 seconds on a CDC6400. The resulting solution, \(y^0\), had positive flow on only 100 of the arcs. These 100 arcs were divided into 5 groups of 20 each and 5 resource constraints were divided so as to make \(y^0\) infeasible. The resource-constrained problem was then solved by the BOXSTEP method for several different box sizes. As in section 5 the best performance was obtained by saving old cuts and omitting the line search. Table 5 contains the results for a series of runs, each starting at the origin. The maximum number of cuts retained was 20 and \(\epsilon=10^{-6}\) was used for each run. 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 25 test problems reported in section 5 were solved using the Dantzig-Wolfe approach. This was done on the B6700 and the counterpart of Table 2 is reported in Table 6.
Table 5. Solution of resource-constrained network problem by BOXSTEP (Dantzig-Wolfe) with varying box sizes.

<table>
<thead>
<tr>
<th>BOX SIZE ($\beta$)</th>
<th>NUMBER OF BOXES REQUIRED</th>
<th>$T$ (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>.1</td>
<td>?</td>
<td>$&gt;150$</td>
</tr>
<tr>
<td>.5</td>
<td>7</td>
<td>70</td>
</tr>
<tr>
<td>1.0</td>
<td>4</td>
<td>55</td>
</tr>
<tr>
<td>2.0</td>
<td>2</td>
<td>76</td>
</tr>
<tr>
<td>3.0</td>
<td>2</td>
<td>119</td>
</tr>
<tr>
<td>4.0</td>
<td>1</td>
<td>92</td>
</tr>
<tr>
<td>5.0</td>
<td>1</td>
<td>108</td>
</tr>
<tr>
<td>1000.0</td>
<td>1</td>
<td>$&gt;150$</td>
</tr>
</tbody>
</table>
Table 6. Solution of 25 resource-constrained network problems by BOXSTEP (Dantzig-Wolfe) with varying box sizes.

<table>
<thead>
<tr>
<th>$\beta$ (box size)</th>
<th>$T_1^*$ (seconds)</th>
<th>$T_2^*$ (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^0$</td>
<td>67.8</td>
<td>3.7</td>
</tr>
<tr>
<td>$10^1$</td>
<td>35.1</td>
<td>4.8</td>
</tr>
<tr>
<td>$10^2$</td>
<td>17.0</td>
<td>9.9</td>
</tr>
<tr>
<td>$10^3$</td>
<td>22.4</td>
<td>14.9</td>
</tr>
<tr>
<td>$10^4$</td>
<td>27.1</td>
<td>17.0</td>
</tr>
</tbody>
</table>

$T_1^*$: 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. Application: Subgradient optimization

A very promising new approach to the class of problems we have been con-
sidering was initiated by Held and Karp [13] in their work on the traveling
salesman problem. This approach, called subgradient optimization, has been
further developed by Held and Wolfe [14] and Held, Wolfe, and Crowder [15].

The problem addressed is

\[(7.1) \quad \max_{y \in Y} v(y) \]

where \(Y\) is convex and \(v(y)\) is defined as in (3.1) as the pointwise minimum of a
family of linear functions.

\[(7.2) \quad v(y) = \min_{k \in K} (f_k + g_k y). \]

If \(v(y^t) = f^k(t) + g^k(t) y\), then it is well known that \(g^k(t)\) is subgradient of
\(v\) at \(y^t\). Subgradient optimization consists of the following iterative process,
starting at any point \(y^1\):

\[(7.3) \quad y^{t+1} = P_Y(y^t + s^t g^k(t)) \]

where \(P_Y\) is the operator projecting \(R^n\) onto \(Y\), \(g^k(t)\) is a subgradient of \(v\) at \(y^t\)
and \(\{s^t\}_{t=1}^{\infty}\) is a sequence for which \(s_t \to 0\) but \(\sum_{t=1}^{\infty} s_t = \infty\). The convergence of
this process is discussed in [14]. We note here only that the sequence \(v(y^t)\) is
not monotonic. In practice it has been found that (7.3) can produce a near
optimal solution of (7.1) very quickly. Achieving complete optimality, which
is important if (7.1) is the dual of the problem of interest, is often consider-
ably more difficult.

Evidently subgradient optimization and the BOXSTEP method would compliment
each other. Subgradient optimization can get close to the global optimum quickly,
and BOXSTEP works best when the global optimum is somewhere in the first not-too-
large box. Any point \(y^t\) produced by (7.3) can be used as the starting point for
BOXSTEP. Furthermore, the subgradients that have been generated provide an initial set of linear supports for BOXSTEP.

This idea for a hybrid algorithm was implemented for the p-median problem (see Marsten [18] or ReVelle and Swain [21]). The continuous version of this problem has the form

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}
\]

subject to

\[
\sum_{j=1}^{n} x_{jj} = p \quad (7.5)
\]

\[
x_{ij} \leq x_{jj} \quad \text{for all } i \neq j \quad (7.6)
\]

\[
0 \leq x_{ij} \leq 1 \quad \text{for all } i, j \quad (7.7)
\]

\[
\sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i = 1, \ldots, n \quad (7.8)
\]

Dantzig-Wolfe decomposition, as developed in section 4, was applied to this problem. Dualizing with respect to the constraints (7.8) results in a Lagrangean subproblem for which a very efficient solution recovery technique has been devised by Blankenship [3]. The starting point for BOXSTEP was obtained by making 250 steps of the subgradient optimization process (7.3). The sequence \(s_t\) was taken as 5 repetitions of 40/t for each \(t = 1, \ldots, 50\). The results are given in Table 7. The column headed \(p\) gives the median sought, as specified in (7.5). The columns headed \(L_{125}^\max\) and \(L_{250}^\max\) give the maximum value of the Lagrangean found during the first 125 and 250 steps, respectively. \(L_{\max}\) is the true maximum value of the Lagrangean. \(T_1\) and \(T_2\) give the time devoted to subgradient optimization and BOXSTEP, respectively, in CDC6400 seconds. The last column gives the number of boxes used during the BOXSTEP phase. This test problem had \(n=33\) and the box size \(\beta=1\) was used in each case. Up to 50 cuts were retained and no line search was used. For \(p=2\) and \(p=4\) an optimal solution
and zero subgradient were obtained very quickly, making the subsequent steps and BOXSTEP solution superfluous. In the remaining cases BOXSTEP, by using more than strictly local information, was able to find and verify the optimal solution after the subgradient optimization process had slowed down. These results are very encouraging and further experimentation is underway with other types of problems.
Table 7. Results for p-median problem with hybrid algorithm

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

The BOXSTEP method replaces a given convex optimization problem with a finite sequence of presumably easier local problems. If each local problem is solved by outer approximation, then BOXSTEP can be viewed as creating an algorithmic continuum between feasible directions methods and cutting plane methods. Like the former, BOXSTEP can take advantage of a good starting point. Like the latter, BOXSTEP can capitalize on accumulated knowledge of the function being optimized.

Our computational work to date has focused on problems obtained by applying Dantzig-Wolfe or Benders decomposition to large specially structured linear programs. The traditional approach to these problems amounts to using one infinitely large box centered at the origin. Our results show that performance can be improved, often dramatically, by using a sequence of "moderate" sized boxes. The meaning of "moderate" is highly dependent on problem structure and scaling but is not difficult to determine for a given problem. 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 reinvestigated. This is especially true when BOXSTEP is considered in conjunction with subgradient optimization as in section 7.

**Structured non-linear programs:** BOXSTEP has yet to be tried on any of the non-linear generalizations of Dantzig-Wolfe or Benders decomposition. See, for example, [8,9].

**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 cutting plane algorithms. Some of the BOXSTEP ideas might be used to enhance the performance of MAP [4, 11, 20] procedures.

**Integer programming:** Geoffrion [10] and Fisher and Shapiro [6] have recently shown how the maximization of Lagrangian functions can provide strong bounds in a branch-and-bound framework. The BOXSTEP method may find fruitful applications in this context. It has the desirable property that the maximum values for successive boxes form a monotonically increasing sequence.

We have also made only the most rudimentary implementation of the method. In all of the experiments reported here we have used a box of fixed size and have insisted upon optimizing each local problem (at least to within $\varepsilon$). Experiments currently underway indicate that better results can be obtained by changing the size and shape of the box at each step and by suboptimizing most of the local problems. This will be reported in a subsequent paper.


<table>
<thead>
<tr>
<th>Date</th>
<th>Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC 19 75</td>
<td></td>
</tr>
<tr>
<td>MAY 12 77</td>
<td></td>
</tr>
<tr>
<td>APR 01 77</td>
<td></td>
</tr>
<tr>
<td>APR 22 77</td>
<td></td>
</tr>
<tr>
<td>JUN 01 77</td>
<td></td>
</tr>
<tr>
<td>JUL 25 77</td>
<td></td>
</tr>
<tr>
<td>AUG 25 77</td>
<td></td>
</tr>
<tr>
<td>FEB 20 79</td>
<td></td>
</tr>
<tr>
<td>MAY 22 00</td>
<td></td>
</tr>
<tr>
<td>DEC 31 00</td>
<td></td>
</tr>
<tr>
<td>DEC 1 00</td>
<td></td>
</tr>
<tr>
<td>DEC 2 00</td>
<td></td>
</tr>
</tbody>
</table>

Lib-26-67