ABSTRACT

The computational difficulties that continue to plague decomposition algorithms, namely, "long-tail" convergence and numerical instabilities, have served to dampen enthusiasm about their computational effectiveness. The use of interior points of subproblems in decomposition procedures may have a significant role to play in alleviating such computational difficulties. Indeed, Dantzig-Wolfe decomposition provides the arena within which simplex techniques for master problems and interior-point techniques for subproblems complement one another in a useful way. In combination they could lead to more effective decomposition algorithms than we have today. We formulate a particular algorithm along these lines and illustrate its convergence and numerical characteristics through numerical experiments. We make these experiments the basis for a discussion of the merits of using interior points in decomposition.

1. INTRODUCTION

In [6], Kantorovich makes the following remarks on decomposition:

A problem that needs to be pointed out especially is that of decentralized decisions. The investigation of a two-level model leads us to the conclusion that in principle the decentralization of decisions with observance of the total objective of the problem is
possible by means of a correct construction of objectives in submodels. We must point out here a brilliant mathematical formalism of the idea of decomposition given by G. Dantzig and P. Wolfe. The value of their paper of 1960 is far greater than the limits of the algorithm they proposed and its mathematical foundation. It gave rise to many discussions and alternative treatments all over the world, and particularly in our country. [Italics ours.]

Dantzig-Wolfe decomposition [3] applied to two-level decentralized decision making has indeed enjoyed dramatic success, and successful application has occurred despite the limitations of decomposition algorithms alluded to above (see [9] and references given therein). In such applications, the underlying linear program is usually of block-angular or dual block-angular form with relatively few linking rows or linking variables and relatively numerous diagonal blocks, and often the blocks have additional structure that can be exploited by using specialized solution techniques.

There has also been a substantial research effort into the application of Dantzig-Wolfe decomposition to time-staged or multilevel decision making, an idea originally suggested in Dantzig [2], and subsequently studied by several authors. Here the underlying linear program normally assumes a staircase structure, and decomposition is applied recursively or “nested.” Perhaps surprisingly, this approach does work, but the performance of nested decomposition algorithms has not been spectacular vis-à-vis the simplex algorithm applied to the original undecomposed problem. For staircase problems, recent interior-point algorithms, spearheaded by Karmarkar’s work [7], have shown substantial promise vis-à-vis the simplex method. It is therefore possible that nested decomposition will become increasingly unattractive as an alternative approach.

Can the same conclusion be drawn about two-level decomposition? We believe the answer is in the negative, for several reasons. First, two-level decomposition has a very natural conceptual and economic interpretation in terms of decentralized decision making. Second, decomposition can take the particular structure of subproblems into account. Third, the approach lends itself very naturally to parallel processing. Fourth, two-level decomposition provides the basis for aggregation techniques and forms the backbone of procedures for areas such as two-stage stochastic programming with recourse, whose particular characteristics make them especially amenable to solution by decomposition techniques; see Ermoliev and Wets [5].

Serious computational difficulties have continued to plague decomposition algorithms in practice—in particular, slow or what is sometimes termed “long-tail” convergence as the optimal solution is approached, as well as numerical instabilities. These have served to dampen enthusiasm about the computational effectiveness of decomposition. We believe that the use of interior points of subproblems in decomposition procedures may have a
significant role to play in alleviating such computational difficulties. Furthermore, it may well turn out to be the case that Dantzig-Wolfe decomposition provides the arena within which simplex techniques for master problems and interior-point techniques for subproblems complement one other in a useful way. In combination, they could lead to much more effective decomposition algorithms than we have today. We enlarge on this theme in the next section, where a specific formulation is discussed. Numerical experiments are described in Section 3, and conclusions and work in progress are briefly discussed in Section 4.

We shall assume familiarity with Dantzig-Wolfe decomposition in its various forms. For example, see Nazareth [10, Chapter 12] for background.

2. INTERIOR POINTS AND DECOMPOSITION

For purposes of our discussion it will be sufficient to consider the following symmetric primal-dual form of the linear programming problem (generalization to other forms is straightforward):

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{s.t.} & \quad A x \leq b, \\
& \quad x \geq 0.
\end{align*}
\]

Partition the foregoing program as follows:

\[
(P): \quad \begin{align*}
\text{minimize} & \quad c^T x \\
\text{s.t.} & \quad A^1 x \leq b^1, \\
& \quad A^2 x \leq b^2, \\
& \quad x \geq 0.
\end{align*}
\]

The Dantzig-Wolfe algorithm [3] applied to the linear program (2) solves in a coordinated sequence a restricted master program and subproblem, each by the simplex algorithm [2, 8]. We shall assume, for convenience, that the subproblem corresponding to the constraints \(A^2 x \leq b^2, \ x \geq 0\) is bounded so as to avoid having to introduce extreme rays into the master program. Extreme points of the subproblem are denoted by \(x^j\).
Let us consider some of the computational difficulties that have plagued the algorithm. Our development is best introduced within the context of a specific example. Therefore, suppose that $x \in \mathbb{R}^3$, the master constraints are two planes (not depicted) that intersect in the line $L$, and the subproblem constraints are inequalities that define the bounded polytope shown in Figure 1.

Assume the objective is such that the optimal solution is at the point $x^*$ in the subproblem facet $f$. Computational difficulties encountered in the Dantzig-Wolfe algorithm are of two types:

(a) Combinatorial: The solution $x^*$ is found as a convex combination of the extreme points $x^A$, $x^B$, and $x^C$. For our example, this looks simple enough, but on a practical problem the corresponding facet $f$ will be of higher dimension and have a complex combinatorial structure. Considerable effort can be expended in the final stages to find the right convex combination, and this is often taken to be the explanation of the slow convergence near the solution. Furthermore, although we have considered here the facet of the subproblem that contains the optimal solution, the same sort of
difficulty, again combinatorial in nature, can occur in earlier stages as the optimal facet of the subproblem is being sought.

(b) Numerical: Some of the vertices that define the optimal solution can be ill conditioned. For example, the vertex $x^A$ could be made to recede very far from $x^B$ and $x^C$ with $x^*$ still lying in the triangle defined by these three vertices. The computed vertex $x^A$ could then be very different from the true one. The corresponding restricted master columns would also in all likelihood have some very badly scaled columns and themselves suffer numerical difficulties. As in the foregoing item (a), these difficulties could also arise prior to arrival at the subproblem facet that contains the optimal solution.

Now there is no reason why $x^*$ must be defined by vertices of the subproblem polytope (here assumed bounded for purposes of discussion), and indeed their use is simply a consequence of the fact that points returned by the simplex method applied to the subproblem are always vertices. The optimal solution $x^*$ could equally well be defined in terms of a convex combination of points that lie within the facet $f$, for example, points $x^P$, $x^K$ and $x^I$. More generally, we propose to derive master columns from points that lie within the interior and on the facets of the subproblem polytope, in addition to extreme points (and extreme rays in the unbounded case). Henceforth such points will be termed grid points. This added freedom can alleviate both of the above difficulties, as the foregoing example illustrates, i.e., the solution $x^*$ can be defined by points that are not dependent on the combinatorial structure of the optimal subproblem facet and would be unaffected by ill-conditioning of some of its vertices. The same holds true for iterates prior to arriving at the optimal solution.

Interior points can be provided by using the Dikin-Karmarkar affine-scaling algorithm [4] or some other interior-point technique to solve the subproblem, and these points can, of course, be arbitrarily close to its facets and vertices. Furthermore, as is well known, it is not at all necessary to minimize the subproblem at each iteration. It is only necessary to satisfy a suitable inequality on the subproblem objective value at the point $x^{k+1}$, the analogue of partial pricing [see (6) below]. Interior-point algorithms are very well suited to this task. We would continue to use the simplex algorithm to solve the restricted master, which one would expect to have relatively few rows and relatively dense columns whose number increases from one cycle to the next. (Each call picks up from the last basis of the preceding call, and the simplex algorithm is especially good at this task.) In contrast, the subproblem(s) would generally be large and sparse and thus suitable candidates for application of an interior-point algorithm. Note that our discussion has been presented in the context of the linear program (2), but it applies equally well
to more general block and dual block angular forms. The algorithm that we propose is as follows:

**Algorithm DKN:**

**Step 1** (Initialize): Choose a set of, say, \( m_1 \) grid points, \( x^1, \ldots, x^{m_1} \), so that the constraints

\[
\sum_{j=1}^{m_1} (A^1 x^j) s_j \leq b^1, \\
\sum_{j=1}^{m_1} s_j = 1, \quad s_j \geq 0
\]

have a feasible solution. Set \( k \leftarrow m_1 \).

**Step 2** (Solve restricted master):

minimize \( \sum_{j=1}^{k} (c^T x^j) s_j : \)

\[
\pi^k : \sum_{j=1}^{k} (A^1 x^j) s_j \leq b^1, \\
\rho^k : \sum_{j=1}^{k} s_j = 1, \quad s_j \geq 0.
\]

Let \( \pi^k \) and \( \rho^k \) denote the associated optimal dual multipliers, and \( s^k_j \) denote the optimal variables of the foregoing restricted master. Let \( \Delta^k \equiv \{ j : s^k_j > 0 \} \).
THE DECOMPOSITION PRINCIPLE

Step 3 (Define current primal feasible solution): Form \( X^k = \sum_{j \in A^k} s_j x_j \).

Step 4 (Define new grid point \( x^{k+1} \)): Define \( \sigma^k = c - (A^1)^T \pi^k \), and consider the subproblem

\[
\text{minimize } (\sigma^k)^T x:
\]

\[
A^2 x \leq b^2,
\]

\[
x \geq 0.
\]

Approximately solve the subproblem using the Dikin-Karmarkar affine scaling algorithm started from the point \( X^k \). For example, ensure that the returned point \( x^{k+1} \) satisfies

\[
\left[ c - (A^1)^T \pi^k \right]^T x^{k+1} - \rho^k \leq -\varepsilon,
\]

where \( \varepsilon \) is a small positive number. Note that the choice \( X^k \) for the starting point ensures that

\[
\left[ c - (A^1)^T \pi^k \right] X^k - \rho^k = 0.
\]

Stop if no further improvement in subproblem objective value is made; otherwise set \( k \leftarrow k + 1 \) and return to step 2.

Suppose that the subproblem is minimized at each call to step 4, with other steps being as above, and let us refer to the resulting algorithm by the acronym DK. Consider also the case when the simplex algorithm is used to solve the subproblem in step 4, again with minimization of its objective function at each call and with starting point given by \( x^k \), the subproblem vertex where it left off at the previous call. Let us refer to this algorithm by the acronym DS. It is clear that DK and DS both generate identical iterates \( X^k \). (Their relative efficiency would depend, of course, on their relative efficiency in solving the subproblem.) DS is an implementable algorithm (terminology of Polak [12]) for which convergence under the usual nondegeneracy assumptions is well known. DK is a conceptual algorithm (again terminology of Polak [12]) because solution of the subproblem is not finite. Convergence of this conceptual algorithm DK follows from convergence of DS and of the Dikin-Karmarkar algorithm under the usual nondegeneracy assumptions.
Of course, the key point is that subproblems are not minimized. Let us refer to the corresponding algorithms by the acronyms DSN and DKN. For algorithm DSN the call to step 4 would continue to return an improving subproblem vertex or terminate. Since the number of subproblem vertices is finite, convergence of the algorithm is still assured in the usual way under nondegeneracy assumptions. (Note that algorithm DSN returns a sequence of iterates $X^k$ that can lie in the interior of the original primal polytope.) Now, however, algorithm DKN would usually return an interior point of the subproblem. This is the new ingredient. The value of the objective function $c^T X^k$ at successive iterates is obviously monotonically decreasing, but convergence of the algorithm must nevertheless be established formally. The situation is analogous in many ways to the use of inexact linesearches in a gradient-based nonlinear programming algorithm. It is not the purpose of this paper to study formal issues of convergence. These and other questions are currently under investigation and will be reported in a subsequent paper. Rather, our objective here is to report on some numerical experimentation with the above algorithm and make these experiments the basis for a discussion of some of the advantages of using interior points in a decomposition setting.

Before concluding this section it is also worth noting that the foregoing algorithms are all special cases of Wolfe’s generalized linear programming (see Dantzig [2] or Shapiro [13]) with exact solution of the Lagrangian subproblem in the case of algorithms DS and DK and with inexact solution of the Lagrangian subproblem in the case of DSN and DKN.

3. NUMERICAL EXPERIMENTS

The algorithms we study are DSN and DKN of the previous section. At step 4 of DKN we permit at most $L$ iterations of the Dikin-Karmarkar affine-scaling algorithm [4] at each call ($L$ will be given below). The latter algorithm is now well known. We made the choice $\alpha = 0.97$ for the factor by which a step along a search direction to the boundary is multiplied in order to bring the next iterate back into the interior. DSN is also outlined at the end of Section 2, and again at most $L$ iterations are permitted at the corresponding step 4.

The algorithms were implemented in a very direct manner in Gauss (Version 1.49B). Our main concern was to obtain initial experience on some numerical examples, and little attention was paid to efficient basis handling in the simplex algorithm and efficient computation of search directions in the Dikin-Karmarkar algorithm. Our numerical experiments are designed to
demonstrate viability of the algorithm DKN and to illustrate the strengths of
our particular approach, and we do not seek here to make comparisons in
terms of efficiency with any other approach.

Two numerical experiments will now be described.

3.1. Illustration of Convergence Characteristics

We utilize the Kuhn-Quandt problems, which have traditionally been
enlisted for such experiments (see Chvatal [1]). These problems are of the
form (1), namely,

\[
\begin{align*}
\text{minimize} & \quad -1^T x \\
\text{s.t.} & \quad Ax \leq b, \\
& \quad x \geq 0,
\end{align*}
\]

where \( A \) is a small dense \( m \times n \) matrix \((m \geq n)\) with integer elements
chosen at random in the range 1 to 100. The numerical range for integer
elements of the vector \( b \) is from 5000 to 10000. This choice will guarantee
that the problems are bounded and that an initial feasible solution is readily
available at the origin. \( 1^T \) denotes a row vector with each element set to
unity.

The coefficient matrix \( A \) is partitioned to form a two-level decomposition
problem of the form (2) with \( c = -1^T \) and such that the first 20 percent of
the rows of \( A \) are considered as the constraints for the master problem and
the remaining 80 percent for the subproblem.

The implementations of DSN and DKN are as described above with
\( L = 3 \). These were first validated by comparing answers obtained on several
Kuhn-Quandt problems with answers obtained by applying the simplex
algorithm to the original (i.e., undecomposed) problems. Then the two
methods DSN and DKN were tested on a number of problems that differed
in their number of variables and equations. We include two illustrative test
problems for the comparison of numerical results (Figures 2, 3). These
figures show how the values of the objective changes after each cycle of the
algorithms and hence the pattern of the convergence to the optimal solution.
(Since the complete sequence is shown, the termination criteria for the
algorithms DSN and DKN are not of importance in our experiments. We
terminated the algorithms with objective-function values found to at least
four significant digits.) A cycle, consisting of a solve of master and subprob-
lem, generally took two simplex iterations in the master and up to three
iterations (simplex or Dikin-Karmarkar) in the subproblem. DKN shows a
similar pattern of convergence to DSN, and one characteristic of DKN was
**Fig. 2.** Kuhn-Quandt problem: Dimension 15. Number of constraints 15.

**Fig. 3.** Kuhn-Quandt problem: Dimension 15. Number of constraints 30.
that it found an approximating solution to the optimal solution at a much earlier cycle. In particular, Figures 2 and 3 show the typical "long tail" that is characteristic of simplex-based decomposition, which could be reasonably terminated around cycle 15. Algorithm DKN, in contrast, could have been terminated at a much earlier cycle. (In addition to these two examples, we ran numerous others, and the behavior shown in Figures 2 and 3 was typical.)

However, one must not read too much into these examples. A single subproblem iteration of DKN usually yields much greater progress but is also much more expensive than a subproblem iteration of DSN. On the other hand, note that a call to the subproblem in DSN usually involves a basis reinversion [costing $O(m_2^3)$ operations, where $m_2$ is the number of subproblem rows.] Also note that updating or use of iterative methods is possible in the Dikin-Karmarkar algorithm applied to the subproblem solved by DKN. Thus the difference in cost between a few subproblem iterations of the simplex algorithm with an initial or final basis reinversion and a few iterations of the Dikin-Karmarkar algorithm may not be as large as might first seem to be the case. Any practical comparison of the two approaches would have to take such considerations into account and is beyond the scope of the current study.

The total number of cycles needed in DSN and DKN to achieve desired accuracy differed from one problem to another, but in small scale problems both methods required approximately the same number of cycles. We expect that the number of iterations and cycles in DKN will become considerably less than for DSN as the size of the linear programming problems becomes larger.

3.2. Illustration of Numerical Characteristics

Here we illustrate the potential advantages of using interior points in decomposition from the standpoint of improving numerical behavior. We consider a problem of the following form, a variant of an example given in Nazareth [10]:

\[
\begin{align*}
\text{minimize} & \quad - x_1 - x_2 - x_3 - x_4 - x_5 \\
\text{s.t.} & \quad x_1 + x_2 + x_3 + x_4 + x_5 \leq 1, \\
& \quad \epsilon x_5 \leq 1, \\
& \quad \epsilon x_5 \leq 1, \\
& \quad \epsilon x_5 \leq 1, \\
& \quad \epsilon x_5 \leq 1, \\
& \quad x_1, x_2, x_3, x_4, x_5 \geq 0,
\end{align*}
\]
TABLE 1

Successive approximations to solution $X^k$

<table>
<thead>
<tr>
<th>$c^TX^k$</th>
<th>$X^1$</th>
<th>$X^2$</th>
<th>$X^3$</th>
<th>$X^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X^k_1$</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>$X^k_2$</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>$X^k_3$</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>$X^k_4$</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>$X^k_5$</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Index  Value

Successive grid points $x^{k+1}$

<table>
<thead>
<tr>
<th>$x^2$</th>
<th>$x^3$</th>
<th>$x^4$</th>
<th>$x^5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>$10^{14}$</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Final matrix for master program

1.0 0.0 0.0 0.0 10^{14}

0.0 1.0 1.0 1.0 1.0

The first column of the master corresponds to a slack variable. The remaining columns correspond to the vertices $x_2,...,x_5$. The rows labelled Index and Value identify the convex combination of vertices that define approximations to the solution $X^k$. For example, $X^4$ is a convex combination of $x_4$ and $x_5$ with corresponding weights $1-10^{-14}$ (rounds to 1.) and $10^{-14}$. If there is only one index as with earlier approximations to the solution, then the slack variable was used in the corresponding basis of the master.
Successive approximations to solution \( X^k \)
\[
e^T X^k = \\
| X^1 & X^2 & X^3 & X^4 & X^5 & X^6 & X^7 |
|---|---|---|---|---|---|---|
| -0.5 & -1.3806 & -4.9905 & -4.9976 & -4.9978 & -4.9999 |
| \( X^k \) & 0.1 & 0.12 & 0.9779 & 0.9977 & 0.99944 & 0.99948 & 0.99998 |
| \( X^5 \) & 0.1 & 0.12 & 0.9779 & 0.9977 & 0.99944 & 0.99948 & 0.99998 |
| \( X^7 \) & 0.1 & 0.12 & 0.9779 & 0.9977 & 0.99944 & 0.99948 & 0.99998 |
| \( X^9 \) & 0.1 & 0.9 & 0.9948 & 0.9997 & 0.99987 & 0.99988 & 0.99997 |
| Index & 2 & 2,3 & 3,4 & 4,5 & 5,6 & 5,7 & 7,8 |
| Value & 1. & 0.9775 & 0.0241 & 0.0789 & 0.9312 & 0.9301 & 0.9983 |
| & .0224 & 0.9759 & 0.9211 & 0.0688 & 0.0699 & 0.0017 |

Successive grid points \( x^{k+1} \)
\[
\begin{array}{cccccccc}
  x^2 & x^3 & x^4 & x^5 & x^6 & x^7 & x^8 \\
  \hline
 0.1 & 0.9992 & 0.9774 & 0.9994 & 0.9994 & 0.9999 & 0.9999 \\
 0.1 & 0.9992 & 0.9774 & 0.9994 & 0.9994 & 0.9999 & 0.9999 \\
 0.1 & 0.9992 & 0.9774 & 0.9994 & 0.9994 & 0.9999 & 0.9999 \\
 0.1 & 0.9992 & 0.9774 & 0.9994 & 0.9994 & 0.9999 & 0.9999 \\
 0.1 & 35.78 & 0.1362 & 1.074 & 0.009 & 0.0188 & 587.68 \\
 0.1 & 0.0008 & 0.0227 & 0.0006 & 0.0006 & 1.47 \times 10^{-5} & 1.30 \times 10^{-6} \\
 0.1 & 0.0008 & 0.0227 & 0.0006 & 0.0006 & 1.47 \times 10^{-5} & 1.30 \times 10^{-6} \\
 0.1 & 0.0008 & 0.0227 & 0.0006 & 0.0006 & 1.47 \times 10^{-5} & 1.30 \times 10^{-6} \\
\end{array}
\]

Final matrix for master program
\[
\begin{array}{cccccccc}
 1. & 0.2 & 35.88 & 0.139 & 1.074 & 0.012 & 0.0189 & 587.681 \\
 0. & 1. & 1. & 1. & 1. & 1. & 1. & 1. \\
\end{array}
\]

The first column of the master corresponds to a slack variable. The remaining columns correspond to the vertices \( x^2, \ldots, x^6 \). The rows labelled Index and Value identify the convex combination of vertices that define approximations to the solution \( X^k \). For example, \( X^5 \) is a convex combination of \( x^5 \) and \( x^6 \) with corresponding weights .9312 and .0688. If there is only one index as with \( X^1 \), then the slack variable was in the corresponding basis of the master.
where $\epsilon$ is a small number ($0 < \epsilon \ll 1$). We choose $\epsilon = 10^{-14}$. The first constraint defines the master, and the remaining four constraints define the subproblem. The polytope corresponding to this subproblem has vertices as follows: in the hyperplane $x_5 = 0$ its vertices correspond to those of the four-dimensional hypercube in the variables $x_1, \ldots, x_4$. In addition, it has vertices at the origin and at the point $(0, 0, 0, 0, 10^{14})$. The optimal solution lies on the edge joining this last point and the vertex $(1, 1, 1, 1, 0)$, and it can be made arbitrarily close to the point $(1, 1, 1, 1, 1)$ by letting $\epsilon \to 0$. Tables 1 and 2 show the sequence of approximations to the solution $X^*$ (step 3), the sequence of grid points $x^{k+1}$ returned by the subproblem (step 4), and the restricted master program (step 2) at termination of algorithms DSN and DKN. The footnote of each table gives further explanation of how the various quantities are defined. In these runs we chose $L = 2$, and the starting points in the subproblem were at the origin for DSN and close to the origin but in the interior for DKN.

In DSN the vertex $x^5$ is essential for defining the optimal solution. The result is an extremely poorly scaled master program with ill-conditioned basis matrices. In contrast, in DKN this type of numerical difficulty does not rise. Our example has been chosen so as to illustrate such difficulties in a blatant manner. For example, DSN could be rectified here by utilizing extreme rays and extreme points in the definition of the master, along with a suitable choice of tolerances in the simplex solution of the subproblem. However, the point is that these difficulties can arise in a practical setting in much more subtle form. At the root of the difficulty is the fact that the combinatorial structure of the subproblem enters into the master and in turn causes numerical difficulties. This can be particular troublesome when removal of some problem constraints that are used to define the master results in a very distorted subproblem polytope with ill-conditioned vertices. In contrast, DKN is much less sensitive to the combinatorial structure of the subproblem polytope.

4. CONCLUSIONS

The decomposition principle of Dantzig and Wolfe is one of the fundamental ideas of mathematical programming. We have sought to show through level-1 experiments (see Nazareth [11] for terminology) that it has subtle and fascinating algorithmic and numerical characteristics, and that the use of interior points may have an important role to play in the formulation of effective algorithms. Work in progress includes the theoretical study of convergence of algorithm DKN, level-2 numerical experiments on prob-
lems more typical of those that arise in practice, and analogues of our results for Benders-type decomposition on the dual program.

REFERENCES


Received 11 December 1989; final manuscript accepted 22 October 1990