A NOTE ON “EFFICIENT NESTED PRICING IN THE SIMPLEX ALGORITHM”

PING-QI PAN

Abstract. Recently, we reported computational results, demonstrating the superiority of a nested pricing rule to other major rules commonly used in practice [8]. This note shows that incorporation of column scaling even enhances the nested pricing rule considerably.

Key words. large-scale linear programming, simplex algorithm, pivot rule, nested, scaling.

AMS subject classifications. 65K05, 90C05

1. Introduction. Consider the linear programming (LP) problem in the standard form

\[ \begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax = b, \quad x \geq 0,
\end{align*} \]

(1.1)

where \( A \in \mathbb{R}^{m \times n} \) \((m < n)\) and \( \text{rank}(A) = m \). It will be a simple matter to extend results of this paper to more general LP problems with bounds and ranges.

The computational efficiency of the simplex algorithm heavily depends on the choice of the pivot rule that is employed to select an entering index, since it essentially determines the number of iterations required for solving LP problems. For this reason, a large number of pivot rules have been proposed, and tested from time to time (for a survey, see [6] or [10]). The steepest-edge rule [3] and its approximation Devex rule [4] are now accepted to be as the best ones, and are commonly used in commercial packages, such as CPLEX [1, 5].

Recently, Pan reported very encouraging computational results with a nested pricing rule [8] against major commonly used rules with 48 largest Netlib problems, all of the 16 Kennington problems and the 17 largest BPMPD problems. This note shows that the nested pricing rule can even be enhanced considerably if a column scaling is simply incorporated in it [9].

2. Modified nested-pricing rule. Let \( B \) be the current basis and \( N \) the associated nonbasis. Without confusion, denote basic and nonbasic index sets again by \( B \) and \( N \), respectively. The reduced costs associated with nonbasic indices may be computed by

\[ \bar{c}_N = c_N - N^T \pi, \quad B^T \pi = c_B. \]

(2.1)

If index set

\[ J = \{ j \mid \bar{c}_j < 0, \ j \in N \}. \]

(2.2)

is nonempty, Dantzig’s original rule [2] selects an entering index \( q \) such that

\[ \bar{c}_q = \min \{ \bar{c}_j \mid j \in J \} < 0. \]

(2.3)

As indicated by [9], the determination of \( q \) is not invariant for scaling. It is seen from (2.1) and (2.3) that quantities \( \pi \) and \( \bar{c}_N \), and hence index \( q \) are all dependent

\*Department of Mathematics, Southeast University, Nanjing, 210096, People’s Republic of China (panpq@seu.edu.cn). Project 10871043 supported by National Natural Science Foundation of China.
of norms of columns of $A$. To eliminated such dependence, all nested pricing rules considered in [8] should be modified. As the nested-Dantzig rule performed best, we only describe an associated variant of it as follows.

**Rule 1 (M-N-Dantzig).** Let an optimality tolerance $\varepsilon > 0$ be given. Set $J = N$ initially.

1. If

$$\hat{J} \triangleq \{ j \mid \bar{c}_j/\|a_j\| < -\varepsilon, \; j \in J \},$$

is nonempty, go to step 4; else,

2. if

$$\hat{J} \triangleq \{ j \mid \bar{c}_j/\|a_j\| < -\varepsilon, \; j \in N \setminus J \},$$

is nonempty, go to step 4; else

3. stop and declare optimality.

4. Determine an entering index $q$ such that

$$q = \arg \min \{ \bar{c}_j/\|a_j\| \mid j \in \hat{J} \},$$

and set $J = \hat{J} \setminus q$ for the next iteration.

It is a simple matter to implement the normalization of columns of $A$ in a scaling preprocess.

**3. Computational Results.** Computational results with Rule 1 turned to be very favorable. We report obtained results in this section.

The following three codes were tested and compared against one another:

1) Devex: Devex rule.
2) MND1: Rule 1 with the 2-norm.
3) MND2: Rule 1 with the 1-norm.

To have the competitions fair and easy, all the three codes were implemented within Minos 5.51 [7] by only changing its rule. Code Devex resulted from Minos 5.51 by replacing its rule by the Devex rule. Codes MND1 and MND2 yielded by inserting a few lines for relevant column normalization in Subroutine m2scla of module M20amat, and using the nested pricing rule.

Our first test set of problems consists of the 47 largest Netlib problems, the second includes all of the 16 Kennington problems, and the third the 17 largest BPMPD problems. These are the same as those used in [8, 9].

From Table 3.1 containing iterations and time ratios of Devex to the new codes, it is seen that both the new codes outperformed Devex for each test set and for all. For the 80 test problems together, MND2 outperformed Devex with total iterations ratio 3.69 and run time ratio 6.08, whereas MND1 defeated Devex with iterations ratio 4.34 and run time ratio 7.27! These results are very favorable, compared with those reported in [8] with their counterpart without the column scaling.

**REFERENCES**

Table 3.1
Ratio Summary

<table>
<thead>
<tr>
<th>Problem</th>
<th>Devex/MND2</th>
<th>Devex/MND1</th>
<th>MND2/MND1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iters</td>
<td>Time</td>
<td>Iters</td>
</tr>
<tr>
<td>Netlib(47)</td>
<td>1.20</td>
<td>1.17</td>
<td>1.15</td>
</tr>
<tr>
<td>Kenningt(16)</td>
<td>5.63</td>
<td>5.42</td>
<td>7.83</td>
</tr>
<tr>
<td>BPMPD(17)</td>
<td>4.32</td>
<td>7.43</td>
<td>5.43</td>
</tr>
<tr>
<td>Average(80)</td>
<td>3.69</td>
<td>6.08</td>
<td>4.34</td>
</tr>
</tbody>
</table>