CHAPTER 3

USING THE GENERAL BLOCKANGULAR BASIS FACTORIZATION (GBBF) IN THE SIMPLEX METHOD

In a revised Simplex Method (see [2]), a representation of the inverse is needed for performing two types of calculations:

1) Solving the system

$$\Pi B_T = C \text{ for } \Pi$$

i.e.
$$\Pi = CB_T^{-1} \tag{3.1}$$

which is computed using the backward transformation (BTRAN) in product form algorithms (see [37]). Here C is the vector of coefficients in the objective function of the basic variables for primal strategies, or the r-th unit vector in dual strategies.

2) Solving the system

$$B_T\bar{d} = d \text{ for } \bar{d}$$

i.e.
$$\bar{d} = B_T^{-1}d \tag{3.2}$$

which is computed using the forward transformation (FTRAN) in product form algorithms. In the section we show how these calculations can be carried out efficiently using GBBF.

3.1 The Backward Transformation (BTRAN)

Using the GBBF representation of the inverse we can write for (3.1)

$$\Pi = CB_T^{-1} = C\hat{V}^{-1}\hat{B}_W^{-1}B_N^{-1} \tag{3.3}$$

Define $\hat{C} = C\hat{V}^{-1}$ and $\hat{\Pi} = \hat{C}\hat{B}_W^{-1}$. We can consider the backward transformation as consisting of three steps:

Step 1: Calculate $\hat{C} = C\hat{V}^{-1}$

Step 2: Calculate $\hat{\Pi} = \hat{C}\hat{B}_W^{-1}$

Step 3: Calculate $\Pi = \hat{\Pi}B_N^{-1}$

which we will now analyze separately.

Let the row vectors $= (C_0, C_1, \ldots, C_k)$, $\Pi = (\Pi_0, \Pi_1, \ldots, \Pi_k)$, $\hat{C} = (\hat{C}_0, \hat{C}_1, \ldots, \hat{C}_k)$ and $\hat{\Pi} = (\hat{\Pi}_0, \hat{\Pi}_1, \ldots, \hat{\Pi}_k)$ be partitioned according to rows in block $0,1,\ldots,k$. Similarly let V_i be the restriction of V to rows in block i. Furthermore, for $i = 1,\ldots,k$ we partition $C_i = (C_i^0, C_i^1)$, $\hat{C}_i = (\hat{C}_i^0, \hat{C}_i^1)$, $\hat{\Pi}_i = (\hat{\Pi}_i^0, \hat{\Pi}_i^1)$, where the subscript 0 corresponds to rows of block i for which the basic variable is in the WB, and the superscript 1 to those basic in their own block. Similarly we partition $V_i = (V_i^0, V_i^1)$ where V_i^0 corresponds to columns basic in the common rows and V_i^1 to the other columns in the Working Basis.

For simplicity let also

$$C_0^1 = (C_1^0,\ldots,C_k^0) \quad \text{and} \quad \hat{C}_0^1 = (\hat{C}_1^0,\ldots,\hat{C}_k)$$

and let I_0^1 be an $\hat{m}$ x $\hat{m}$ identity where $\hat{m}$ is the dimension of C_0^1.

3.1-1 Step 1

By rearranging rows and columns (if necessary) we can express $\hat{C} = C\ \hat{V}^{-1}$ as

$$(\hat{C}_0,\hat{C}_0^1,\hat{C}_1^1,\ldots,\hat{C}_k^1) = (C_0,C_0^1,C_1^1,\ldots,C_k^1)$$

$$\begin{pmatrix} I_0 & & & \\ & I_0^1 & & \\ -V_1^0 & -V_1^1 & I_1^1 & \\ & & & \ddots \\ -V_k^0 & -V_k^1 & & I_k^1 \end{pmatrix}$$

from which we obtain the following relationships.

$$\hat{C}_0 = C_0 - \sum_{i=1}^{k} C_i^1\ V_i^0$$

$$\hat{C}_0^1 = C_0^1 - \sum_{i=1}^{k} C_i^1\ V_i^1 \qquad (3.4)$$

$$\hat{C}_i^1 = C_i^1 \qquad i = 1,\ldots,k \ .$$

Now we have to consider three cases.

Case 1 : During Phase 2

Here $C = (1,0,0,\ldots,0)$, (assuming the objective function is the first row in the common rows) and hence $C_i = 0$ $i = 1,\ldots,k$ Thus from relationships (3.4) we obtain $\hat{C} = C$ and hence Step 1 is not required during Phase 2.

Case 2 : Phase 1, variables not in WB feasible

Again $C_i^1 = 0$ for $i = 1,\ldots,k$ and hence $\hat{C} = C$ and Step 1 is not required.

Case 3 : Phase 1, some variables not in WB infeasible

In this case in general $\hat{C} \neq C$ and we have to go through Step 1. However, if we are minimizing an unweighted sum of infeasibilities, then the components of C take on values 0, 1 or -1*, and hence, as can be observed from relations (3.4) no multiplications or divisions will be necessary, only additions or subtractions.

3.1-2 Step 2

Recall that $\hat{\Pi} = \hat{C}\ \hat{B}_w^{-1}$. This is an ordinary backward transformation, and hence the number of operations required for its calculation will be proportional to the number of non-zeros in the representation of the inverse of the Working Basis.

---

* The infeasibility form can be expressed as $\min \left(\sum_{i \in S_1} X_i - \sum_{i \in S_2} X_i \right)$ where $S_1 = \{i : X_i > 0$ basic and artificial$\}$ and $S_2 = \{i : X_i < 0$ basic$\}$. Thus the components C_i take on values 1, -1 or 0 according to i belonging to S_1, S_2 or none of them (see also [2]).

3.1-3 Step 3

Rewriting $\Pi = \hat{\Pi} B_N^{-1}$ as $\Pi B_N = \hat{\Pi}$ then from the structure of B_N (see (1.1)) we see that

$$\Pi_0 = \hat{\Pi}_0$$

and

$$\Pi_0 A_i + \Pi_i B_i = \hat{\Pi}_i \qquad \text{for } i = 1,\ldots,k$$

or

$$\Pi_i = (\hat{\Pi}_i - \Pi_0 A_i) B_i^{-1} \tag{3.5}$$

or

$$(\Pi_0,0,\ldots,0,\Pi_i,0,\ldots,0) = (\hat{\Pi}_0,0,\ldots,0,\hat{\Pi}_i,0,\ldots,0)\hat{B}_i^{-1} \tag{3.6}$$

That is, for any $i = 1,\ldots,k$, calculating Π_i is equivalent to an ordinary backward transformation using the representation for the inverse of its block basis, and hence the number of operations is also proportional to the number of non zeros in this representation.

3.2 The Forward Transformation (FTRAN)

Using the GBBF we can write for (3.2)

$$\bar{d} = \hat{B}_T^{-1} d = \hat{V}^{-1}\ \hat{B}_w^{-1}\ B_N^{-1} d \tag{3.7}$$

if the incoming column belongs to block i, then by (1.4)

$$B_N^{-1} d = \hat{B}_i^{-1} d \quad \text{and} \quad \bar{d} = B_T^{-1} d = \hat{V}^{-1}\ \hat{B}_w^{-1}\ \hat{B}_i^{-1} d \tag{3.8}$$

That is, we need only its own block inverse, the Working Basis inverse and the V matrix to perform the calculations of the forward transformation.

3.3 Implications on the Choice of Simplex Strategy*

By using the factorized representation of the inverse and by taking full advantage of the structure appreciable savings can be obtained in the number of operations that have to be performed and in the amount of data required in both the backward and forward transformations. In particular:

1) Whenever all blocks are feasible the V matrix is not used in the backward transformation. Hence a good strategy would be to make all blocks feasible in the beginning.

2) When using a primal simplex strategy with partial pricing (see [30]) to coincide with columns in a block (or in some blocks) only the Π_i's corresponding to that block (or blocks) have to be calculated, with considerable savings in the backward transformation with respect to the case when a general representation is used for B_T^{-1}. When using a dual simplex strategy the whole pivot row has to be updated ("priced out") and hence partial pricing is not possible. However there is one special case when using GBBF which is formalized in the following lemma.

---

* Strategy is used here as defined in section 1.2, i.e. rules as to how to iteratively move from one basic solution to the next.

Lemma 3: If the outgoing variable belongs to block i and corresponds to row r of its block basis inverse, and if the corresponding row of V, $v_r = 0$, then the solution to the system $\Pi^r B_T = U_r$, where U_r is the r-th unit vector, is given by

$$\Pi^r_j = 0 \qquad \forall j \neq i$$

and

$$\Pi^r_i = U_r \hat{B}_i^{-1} \quad .$$

Proof: $\Pi^r = U_R B_T^{-1} = U_r \hat{V}^{-1} \hat{B}_w^{-1} B_N^{-1}$. But $U_r \hat{V}^{-1} = U_r$ since $v_r = 0$ and $U_r \hat{B}_w^{-1} = U_r$ since pivot row is not in WB. Thus $\Pi^r = U_r B_N^{-1}$ or equivalently $\Pi^r B_N = U_r$, from which the lemma follows because of the special structure of B_N (see (1.1)). ||

Hence, for updating the whole pivot row we need in this case only to perform one backward transformation using only one block inverse, and to price out only the columns on that block and the columns of the coupling variables, since all the others will price out to 0. The above is a generalization of a result of Ohse [28].

3) Except for the coupling columns the forward transformation also requires only the use of one of the block inverses. Also the updating of the factorized representation of the inverse simplifies considerably in the absence of coupling columns (see Fig. 2 in section 3.4). This suggests a special treatment for coupling columns.

4) Savings occur also in total time spent in inversions,

since on the average smaller matrices will be inverted and the number of inversions not counting those of the Working Basis will remain roughly the same.

With these points in mind we first turn to develop an efficient strategy for the block-angular linear problem with coupling constraints only.

3.4 A Strategy for Block-Angular Linear Problem with Coupling Constraints

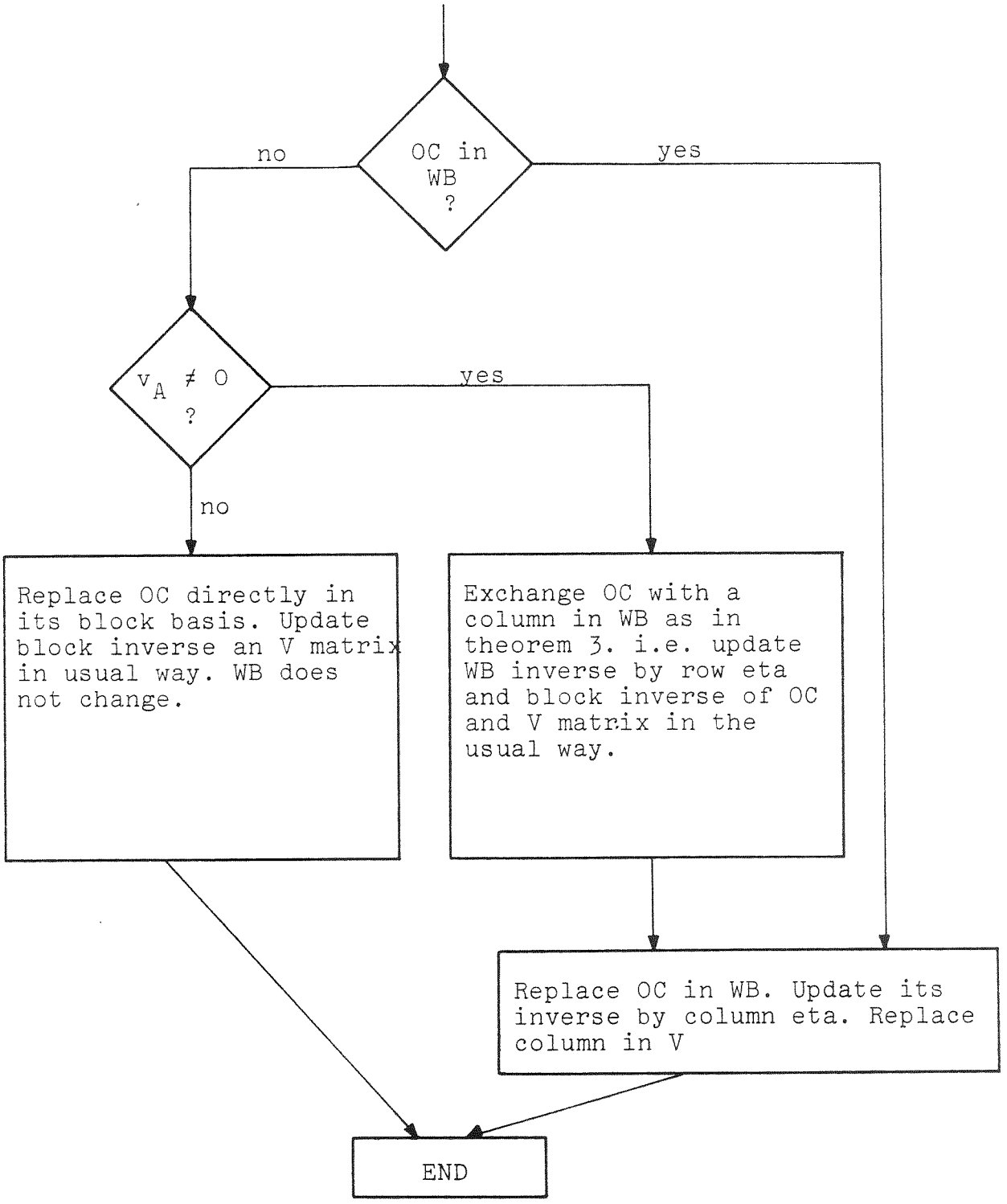
In contrast to block columns, coupling columns require the use of all block inverses both for the backward and for the forward transformation. They also tend to increase the size of the Working Basis and to complicate the updating procedure. For all these reasons they should be treated differently: For instance, to consider them as candidates to enter the basis and to price them out only when no improvements can be made with block variables, or to treat them as fixed parameters whenever possible. Hence the strategy for the special case of a block-angular problem with coupling constraints will play an important role in that of the more general problem P.

3.4-1 Simplifications in the Updating Procedure

Since there are no coupling columns in this case all variables are of type A and the updating procedure given in Fig. 1 reduces to that in Fig. 2. There are now only three updating cases which depend exclusively on the position of the outgoing variable in the basis. Also since the Working Basis includes

FIGURE 2

Information Flow-sheet of Updating Procedure for Block-angular Linear Problems with Coupling Constraints



all common rows, and as a consequence of Lemma 2, for this case we have that the Working Basis will have constant size $m_0 \times m_0$.

3.4-2 Strategy Considerations

Dual and Parametric strategies for block-angular linear problems with coupling constraints (see section 4.1) are based on the observation that if all blocks are independently sub-optimized at the beginning then the resulting solution is dual feasible for the overall problem. Hence the first step in these methods is to sub-optimize all blocks.

On the other hand, in order to make full use of the reduction in computations and data transfer in the backward transformation for primal strategies when using GBBF, it is necessary to have all blocks feasible.

Alternatively we could sub-optimize them on the heuristic that later we could approach feasibility in the common rows "from above" (in the maximizing case) and could expect to arrive at the feasible region with a higher value of the objective function and hence would have fewer iterations in Phase 2. Primal strategies offer the following advantages during this first step:

1) They do not require a dual feasible solution after solving all blocks, allowing us to stop before reaching optimality if this is considered convenient to save iterations.

2) For the same reason they do not require any special treatment if some block has an unbounded solution or if the matrix $D_0 \neq 0$ (see problem P in 2.1).

After this first step primal strategies have the advantage of allowing savings in BTRAN if we make use of partial pricing. Since for large problems partial pricing is desirable to reduce overall computation time, the added benefits of reducing the amounts of data and of computations required in the backward transformation make it even more attractive here. We will refer to the use of the partial pricing technique applied to the columns of a block as the Partial Block Pricing strategy, or PBP for short.

The above considerations lead to the following two step primal strategy:

Step 1: Optimize all block problems (alternatively stop once feasibility is reached). If some block has no feasible solution STOP, the whole problem is infeasible. Otherwise proceed to Step 2.

Step 2: Use the Partial Block-Pricing strategy to take advantage of the savings in BTRAN that are made possible by the factorized representation of the inverse. This step terminates in one of the usual primal termination states, i.e. no feasible solution, unbounded solution or optimal solution.

3.4-3 Experimental Results

The above two step strategy for block-angular linear problems with coupling constraints has been coded in FORTRAN IV in a program called G-GUB. Preliminary experimental results look

promising and are reported in [38]. In Appendix A we give a more detailed flow-sheet of the algorithm resulting from the application of the above two-step strategy and reproduce some of the results in [38].

In the following we will refer to this algorithm as the Coupling Constraints Algorithm (or CCA for short). For other strategies for block-angular linear problems with coupling constraints see section 4.1.

3.5 A Strategy for the General Problem (P)

Pricing out and Updating a coupling column requires in general the use of k block inverses instead of one. They also tend to increase the size of the Working Basis and to complicate the updating of the factorized representation of the inverse. To amplify these points further::

a) From Lemma 2, if there are m_B coupling variables in the basis then $m_0 + m_B$ is an upper bound on the size of the Working Basis. Thus, the fewer coupling variables in the basis, the smaller is this upper bound estimate.

b) When the size of the Working Basis increases, then so does the number of columns in the V matrix. Moreover, each Type B column (coupling column) can have non-zeros in all rows of V as compared to Type A2 columns which can have non-zeros only in rows of V belonging to their own block. Thus, the higher the number of coupling variables in the basis,

the higher the number of operations required to update a vector.

c) A comparison of figures 1 and 2 shows that the work to update the factorized representation of the inverse can be expected to be more substantial when there are coupling variables in the basis.

Thus we conclude:

1) Pricing out and updating a coupling column to introduce it to the basis requires more work than for a block column, since the full Π vector is needed for pricing and hence all block inverse have to be used during the backward and forward transformations.

2) Having coupling columns in the basis increases the work per iteration even when pricing out only block variables by a), b) and c) above.

From these considerations evolves the general philosophy "do not touch the coupling columns until it becomes necessary". That is, reduce the general problem P to one with only coupling constraints by considering the coupling variables y fixed at some value, and use CCA to solve it. Only then relax the restrictions on y.

The hope is that most of the work can be done without using the coupling variables and that they will enter the game only at the end for relatively few iterations. This is probably the case in many large applications, where from knowledge of the problem it is possible to specify a value of y for which the

whole problem has a feasible solution, and then all of Phase 1 and most of Phase 2 can be done using CCA.

3.5-1 General Strategy

Repeatedly, in the General Algorithm that will be presented in 3.5-2, the values of y will be fixed as a parameter to reduce problem P to one with coupling constraints only. Whenever it is not possible to keep the values of y fixed anymore we will use the following General Strategy, or GS for short:

Step 1: Relax restrictions on y. If some component of y was fixed at some feasible value different from its bounds, introduce it to the basis by increasing its value if this improves the value of the objective function, or by decreasing it otherwise, using the usual Primal Simplex criteria to determine the outgoing variable.

Step 2: Optimize the objective function using the PBP strategy to select the incoming variable (for this purpose consider coupling columns as a block k+1).

3.5-2 A General Algorithm

All the previous consideration lead to the following General Algorithm, whose flow-sheet is given in Figure 3:

Step 0: Fix the coupling variables at a value $y = y_0$. If a value of y is known for which the whole problem is feasible it can be used as y_0. Otherwise an arbitary value between its bounds can be taken.

Step 1: Minimize on each block the sum of infeasibilities. If

all problems get feasible go to Step 3. Else continue to Step 2.

Step 2: Use the General Strategy to minimize the sum of infeasibilities for blocks (common row constraints still relaxed). If no feasible solution is attained STOP, the whole problem is infeasible. Else fix y at its current value and continue to Step 3.

Step 3: Use the Coupling Constraints Algorithm to minimize the sum of infeasibilities in the common rows. If a feasible solution is attained go to Step 5. Else continue to Step 4.

Step 4: Use the General Strategy to minimize the sum of infeasibilities in the common rows. If no feasible solution is attained STOP, the whole problem is infeasible. Else fix y at its current value and continue to Step 5.

Step 5: Use the Coupling Constraints Algorithm to minimize the objective function. If an unbounded solution is encountered STOP, problem unbounded. Else continue to Step 6.

Step 6: Use the General Strategy to minimize the objective function. If an unbounded solution is encountered STOP, problem is unbounded. Otherwise the optimal solution is obtained.

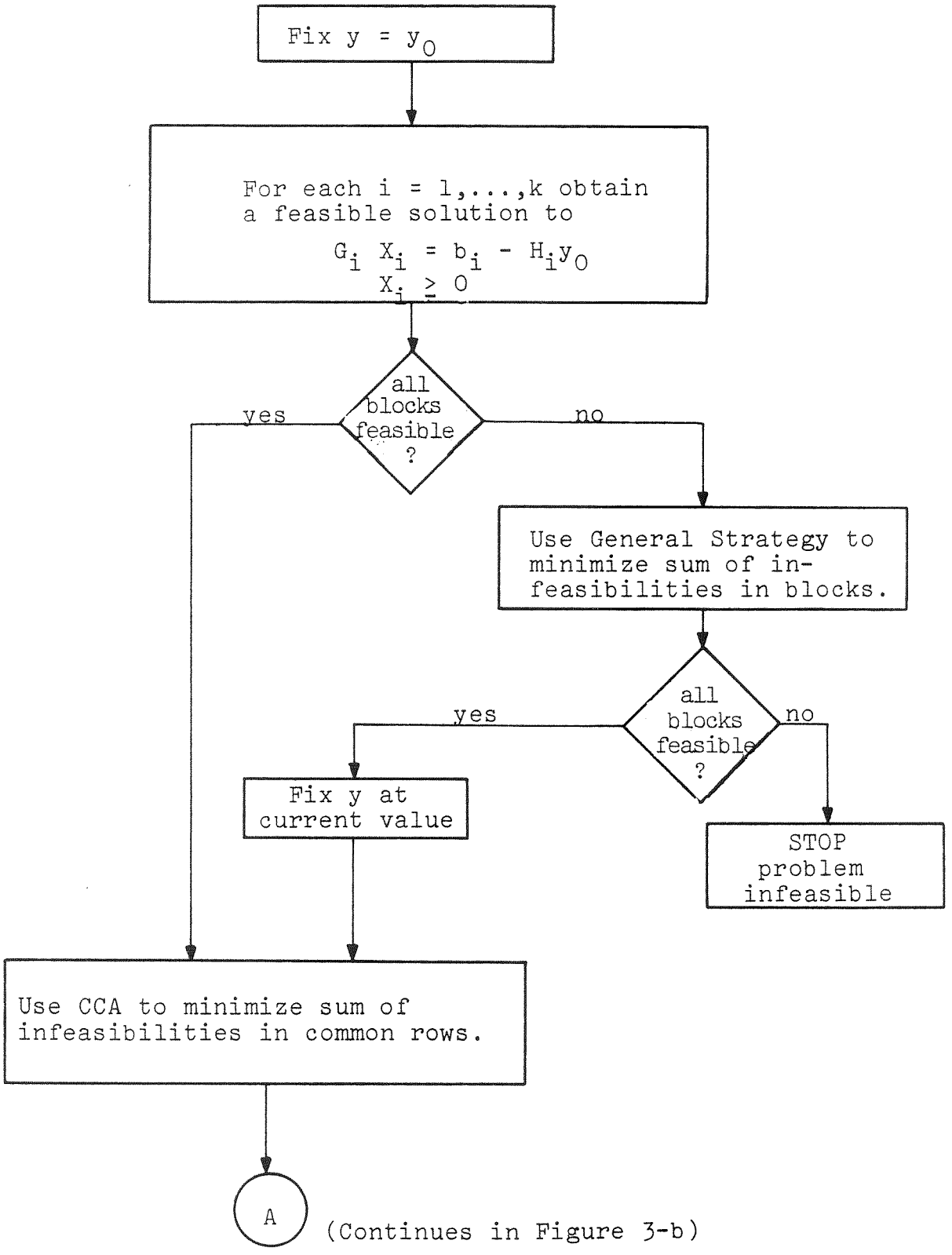
3.5-3 Observations

The General Algorithm has been stated in terms of a General Strategy and the Coupling Constraints Algorithm with the following ideas in mind:

1) The General Strategy as stated in 3.5-1 is one sensible strategy that takes advantage of the structure of the problem by using the Partial Block

FIGURE 3-a

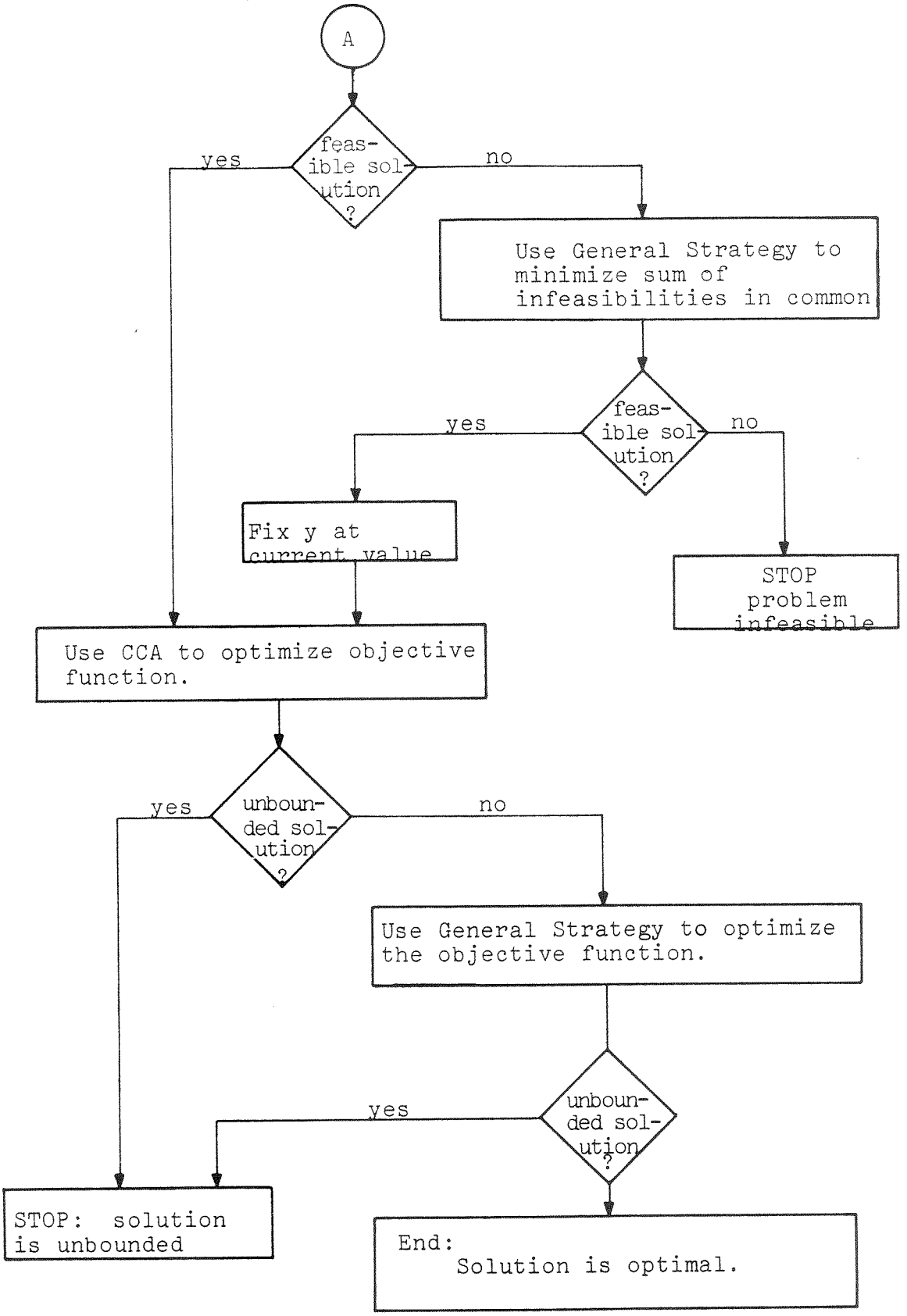
Information Flow-sheet for General Algorithm



(Continues in Figure 3-b)

FIGURE 3-b

(Continuation of Flow-Sheet in Fig. 3-a)



Pricing strategy. If it turns out that better results can be obtained by using a different strategy, like (for example) pricing out coupling variables only when block variables do not supply a candidate, or every t cycles for t > 1, then it can be used as General Strategy to the benefit of the General Algorithm without any other modification.

2) Similarly, if an improved strategy is found for the block-angular linear case with coupling constraints, it can be adopted for the CCA to the advantage of the General Algorithm.

3) Notice that if a y is known for which the whole problem P is feasible, then the method reduces to the Coupling Constraints algorithm followed by Step 6. Thus, when there are no coupling variables it reduces to CCA.

At the end of Chapter 4, after analysing other partitioning and decomposition methods, further strategies are compared.

3.6 Representation for Inverse and V Matrix

So far only some product form representation of the inverse was assumed in describing the Updating of the Factorized Representation of the Inverse. This assumption was not necessary, it was used only because product form representations have been found to be the most efficient ones for general large scale linear problems, and this allowed us to speak of the updating of an inverse in terms of adding a column eta or a row eta to its representation, from which it is easy to obtain an intuitive

feeling of the relative effort required to maintain and to carry an inverse.

Thus for each basis we can use the representation that gives the best results for its inverse. For general sparse basis' an L-U factorization inversion with the use of the Forrest-Tomlin updating method for the triangular factors (see [14]) seem to give the most economic representation, and hence it seems to be the one to use for block inverses.

In certain cases though, some or all of the blocks may have special structure which we can take advantage of. For instance, if some block corresponds to a network, each basis will be a tree and we do not have to keep an inverse, it suffices to keep a set of pointers that allow us to reconstruct the tree [7]. That is, the GBBF approach does not only give as the advantages of a reduced BTRAN and FTRAN, but also allows us to take advantage of the special structure of each block.

The Working Basis can be expected in general to be more dense than the original matrix. Also its size may vary from one iteration to the next and both row and column elementary matrices may be required to update its inverse. Hence the Forrest-Tomlin method can not be used. If the Working Basis is very dense an explicit inverse (see [7]) will be best. Otherwise an L-U factorization followed by product form updates can be used. For the latter the following conditions will keep the density from getting too high:

1) A fair proportion of vectors in the Working Basis can be expected to consist of common row slacks and of

other columns in D_O (see problem P).

2) Common rows will consist of rows with non zeros in more than one block, but not necessarily in all. Columns belonging to a block having only zeros in a common row do not contribute in that row a non zero to the Working Basis. The same is true if all columns of a block having non-zero in a given common row are non basic, and even if some are basic, there is a certain probability that it may still be so.

3) From Theorem 1 for a minimal Working Basis $\hat{U}_A = 0$.

Besides these three points there are also some programming considerations that make it advantageous to use a product form representation for the Working Basis inverse: It is possible to use essentially the same INVERT, BTRAN and FTRAN subroutines for Working Basis and block inverses.

As for the V matrix, from (2.11)

$$\begin{pmatrix} B_w \\ \\ V \end{pmatrix} = B_N^{-1} B_{wO}$$

Hence it is not mandatory to store V, since in B_N^{-1} and B_{wO} we have all the information needed to carry out the computations involving V. In BTRAN V is not needed anyhow when all the variables not in the Working Basis are feasible, and so in this case it would make no difference if we have stored V or not. In FTRAN on the contrary all the block inverses would be required

in the last part of it which involves V. This is only justified in case that the number of non zeros in all block inverses is less than that in V. This would probably mean that the size of the Working Basis is large, since the number of columns in V equals that in the Working Basis, and the advantages of using the GBBF approach are reduced with respect to a general method. Hence for problems where it is advantageous to use the GBBF approach, i.e. those leading to Working Basis with a relatively small size, it is better to store V (i.e. store the non-zeros of V).

Recall that in updating an incoming column d we calculate first

$$\hat{d} = \begin{pmatrix} \hat{d}_w \\ \\ \hat{d}_s \end{pmatrix} = B_N^{-1} d$$

where $\hat{d}_w$ is the restriction of $\hat{d}$ to rows in the Working Basis. Since this information is already available it seems convenient to store the whole updated vector $\hat{d}$. This way we also have available the vectors forming the Working Basis and it is not necessary to recompute them every time we want to invert it. All that is required is a flag that tells us which rows are in the WB and which ones are not.

In cases where storage restrictions do not allow to store V, in addition to a higher computational effort in FTRAN it becomes necessary to generate the row of V corresponding to the

pivot row and which is required in the updating procedure. As can be readily seen from (2.11) this is equivalent to BTRAN using the block inverse of the outgoing variable plus pricing out the columns in the Working Basis with the Π^r thus calculated.

3.7 Other Considerations

Up to now we have implicitly assumed the use of the most negative reduced cost as the criterium for selecting the column to enter the basis among those that were priced out. Of course other criteria and techniques widely used in the primal Simplex Method are also possible here. So for instance multiple pricing [30], i.e. where at each pricing operation the k columns having the most negative reduced cost among those priced out are selected as candidates and updated, and which are then used in a sub-optimization where the greatest change rule is applied to determine the column to enter the basis.

There are also other criteria which do not look good in general, but look promising when part of a GBBF approach. These will be examined in more detail in Chapter 4, especially in Section 4.1-5.

CHAPTER 4

A UNIFYING APPROACH TO PARTITIONING AND DECOMPOSITION METHODS

General Block-angular Basis Factorization allows us to unify existing Partitioning and Decomposition methods (not based on the Dantzig-Wolfe decomposition principle) for solving block-angular linear problems. In essence all of them can be viewed as the Simplex Method using the GBBF representation for the inverse, and differing on the strategy as to the vector pair selected to enter and to leave the basis.

For each method we will refer to the appropriate place in the literature for its detailed description, and state it here only in terms of the pivot strategy it uses. When necessary we will expand somehow on alternative ways of implementing them. We will first consider block-angular linear problems with coupling constraints, then with coupling variables and finally with both coupling constraints and variables.

4.1. Block-Angular Linear Problems with Coupling Constraints

As was discussed in 3.4-1, in this case the Working Basis is always of constant size $m_0 \times m_0$, and the updating procedure for the factorized representation of the inverse

simplifies to that in Figure 2 because of the absence of Type B variables. Many "Partitioning" or "Decomposition" methods have been proposed over the years for this class of problem. We will now analyze them to identify their strategy in the Simplex Method using GBBF (but not necessarily in the order in which they were first presented). At the end of the section we look at some new strategies which can be implemented efficiently when using the GBBF approach in the Simplex Method.

4.1-1 Primal Simplex Strategy: Methods of Kaul [22], Bennett [4] and Müller-Mehrbach [27].

All three authors proposed their methods independently about the same time. Kaul's method is better known as Generalized GUB, (short for Generalized Generalized Upper Bounding [22]) and Müller-Mehrbach's as the Method of Direct Decomposition [27]. There are slight variations in their methods, but all correspond to the GBBF Simplex Method using the usual primal strategy of introducing into the basis the column with the most negative reduced cost, and hence leading "to the same solution path as the Simplex Method" [22], [27].

4.1-2 Rosen's Primal Partitioning Method [32]

We can distinguish three steps in the overall strategy of Rosen's method:

Step 1: Solve all block problems to optimality. If some block is infeasible STOP, the whole problem is infeasible. Else go to Step 2.

Step 2: Relax the non-negativity constraints on variables in the block basis'. Solve the relaxed problem. If solution infeasible STOP, the problem is infeasible. Else go to Step 3.

Step 3: Check if the relaxed non-negativity constraints are satisfied. If so STOP, the solution is optimal. Else rearrange variables between blocks and Working Basis as shown in Rosen [32] for at least one block having a variable not satisfying the non-negativity constraints. This way at least one infeasible variable is exchanged to the Working Basis. Return to Step 2.

Observations:

1) The validity of the above strategy was proven by Rosen in [32]. It follows also from the relaxation strategy in Geoffrion [16] (see also Lasdon [24]), of which this is a special case.

2) Notice that the relaxed problem in Step 2 could be unbounded even though the whole problem is not. To avoid this a bounding row making the sum of all variables less or equal to a very large number is added to the common rows. If this constraint is binding in an optimal solution the whole problem is declared to be unbounded.

3) Notice that at the end of Step 2 the solution is dual feasible for the whole problem and after returning from Step 3 at least one previously violated non-negativity constraint is enforced. Thus the solutions in Step 2 form a non decreasing sequence of lower bounds to the problem (assuming we are minimizing).

4) Rosen [32] suggests using a primal strategy in Step 2. In this case we can take advantage of the savings in BTRAN with the PBP strategy. A primal approach also allows us to handle the situation when some block problem is unbounded without adding a bounding constraint to it. Other authors have suggested using a dual approach (see Grigoriadis [17]) in which case PBP could not be used, and a bounding row would have to be added to any unbounded problem.

5) Notice that all the pivoting in Step 2 occurs in the common rows, i.e. the outgoing column always belongs to the WB, leading to the easiest update situation (see Fig. 2). This is a consequence of relaxing the non-negativity constraints on variables in the block basis'. It also implies that it is not necessary to completely update the incoming column on these rows That is, the V matrix is not required in the forward transformation.

6) The dual form of Rosen's method is known as Gass' Dualplex Method (see 4.2.-2).

4.1-3 Primal-Dual Strategy: Balas [1], Knowles [23]

In this approach all block problems are first optimized. This solution is then dual feasible for the whole problem, and primal feasible except possible in the common rows. The Primal-Dual strategy is then employed to reduce the sum of

infeasibilities in the common rows, maintaining a dual feasible solution for the whole problem.

This strategy is intuitively appealing. Its computational advantages will depend on how many non-basic columns there are on each restricted primal, since if there is only one, the effort per iteration is essentially the same as in a dual method. On the other hand, if there are many, most of the work can be expected to be on the primal iterations of the restricted problem, and in this case we can take advantage of the savings in BTRAN with PBP.

Balas first presented this algorithm as "An Infeasibility Pricing Decomposition Method for Linear Programs (Version A)", [1]. Knowles [23] later wrote a FORTRAN code for a version of the Algorithm. He obtained encouraging preliminary experimental results.

4.1-4 Dual and Parametric Strategies: Ohse [28], Orchard-Hays [29]

Again all blocks are optimized first in order to obtain a dual feasible solution to the whole problem which is primal infeasible only in the common rows. A constraint bounding the sum of the variables to be less or equal to a very large number has to be added in case of unbounded subproblems.
Ohse's Dual Method [28] then uses the usual dual strategy (see [7]), taking advantage of the reduction in computations when using the factorized representation of the inverse whenever the outgoing variable is basic in some block and $v_r = 0$ (see Lemma 3).

Any dual partial pricing scheme to select the outgoing variable can be used. This allows some leverage as to what variable to select to leave the basic set. Recall that the computational effort to update the factorized representation of the inverse depends exclusively on the position of the outgoing variable (see 3.4-1), and hence in this method we can exert some influence to fall into the easier cases.

Orchard-Hays Block-Product Algorithm [29] is a parametric method. It modifies the right hand side of the common rows to make the dual feasible solution obtained after solving all block problems also primal feasible. Then the right hand side of the common rows is varied parametrically to its original value.

Besides adding bounding constraints to handle unbounded blocks, Orchard-Hays also shows how to get a dual feasible solution when having non-unit vectors in D_0 (see [29]). This same approach can be applied to the Primal-Dual and Dual methods when necessary.

4.1-5 Other Strategies

Among many other possible strategies we would like to mention two variations on primal strategies.

The Partial Block Pricing (PBP) primal strategy: referred to already in section 3.4-2 and used in the Coupling Constraints Algorithm which has given encouraging results on some test problems (see Appendix A). All block problems are first optimized (or made feasible). Then the PBP primal Strategy is used to take advantage of the savings in the backward

transformation that it makes possible when using the GBBF method.

PBP with a Ratio Pricing Criteria: Ratio Pricing was proposed originally by Markowitz (see Dantzig [7]). It requires taking for each non-basic column a ratio of its updated coefficients in the objective row and in the infeasibility row. Thus in the usual product form methods two backward transformations, one to obtain the prices on the objective row, the other on the infeasibility row, and two pricing operations, one to update the coefficients in the objective row, the other those in the infeasibility row, are necessary. This makes the computational requirements excessive.

In this context a somehow modified ratio pricing rule is proposed, which can be implemented efficiently on block-angular problems when the GBBF approach is used.

Suppose that in Phase 1, for column j, d_j is the reduced cost for the minimization of the sum of infeasibilities, and c_j for the objective function. Then define

$$SD = \{j : d_j < -\delta\} . \tag{4.1}$$

i.e. the set of columns that would decrease the sum of infeasibilities if introduced to the basis (δ is the 0 tolerance for the reduced cost in the computer).

The ratio pricing criteria suggested by Markowitz is:

$$\text{choose} \quad \frac{c_q}{d_q} = \min \{\frac{c_j}{d_j} : j \in SD\} \tag{4.2}$$

i.e. Choose that column to enter the basis which gives the maximum improvement in the objective function per unit decrease in the sum of infeasibilities.

When solving block-angular linear problems with coupling constraints, after optimizing all block problems, we then have a solution which is dual feasible (unless some block gives an unbounded solution) for the overall problem and primal infeasible only in the common rows. Thus the objective function will have a value below the optimum (minimizing case). Hence we modify the ratio pricing criterium. Herewith define

$$SM = \{j : c_j < -\delta \quad , \quad j \in SD\} \tag{4.3}$$

$$SO = \{j : |c_j| \leq \delta \quad , \quad j \in SD\} \tag{4.4}$$

$$SP = \{j : c_j > \delta \quad , \quad j \in SD\} \tag{4.5}$$

Then in Phase 1, if SD is empty the problem is infeasible. Otherwise use the :

Ratio Pricing Criterium

Rule 1: If SM is empty use rule 2. Otherwise select the incoming variable so that

$$\frac{d_q}{c_q} = \max \{\frac{d_j}{c_j} : j \in SM\} \tag{4.6}$$

i.e. maximize the reduction in the sum of infeasibilities per unit improvement in the objective function.

Rule 2: If SO is empty use rule 3. Otherwise select the incoming variable so that

$$d_q = \min \{d_j : j \in SO\} \tag{4.7}$$

Rule 3: Select the incoming variable so that

$$\frac{c_q}{d_q} = \max \{\frac{c_j}{d_j} : j \in SP\} \tag{4.8}$$

That is, if possible we select using rule 1 the column that gives the highest decrease in the sum of infeasibilities per unit improvement in the objective function. If there is no column which improves both objectives then by rule 2 we select the one that gives the greatest decrease in the sum of infeasibilities without affecting the objective function value. If this set is empty we fall back on rule 3 and select to enter the basis the column that gives the minimum increase in the objective function (minimizing case) per unit decrease in the sum of infeasibilities.

Observe that these pricing criteria have many points in common with the Primal-Dual strategy, but they do not require to keep a dual feasible solution and hence allow the use of partial pricing.

Herewith we change SD to

$$SD_i = \{j : d_j < -\delta \quad , \quad \text{column } j \text{ in block } i\} \tag{4.9}$$

in (4.3) through (4.5). Then if we are in Phase 1 and SD_i is not empty we use the ratio pricing criterium. If SD_i is empty

we proceed to another block. If it is empty for all blocks i then the problem is infeasible. Now we can state the :

PBP Algorithm using the Ratio Pricing Criterium:

Step 1: Optimize each block problem (or make feasible). If some problem is infeasible STOP, the overall problem is infeasible.

Step 2: Minimize the sum of infeasibilities in the common rows using the Partial Block Pricing strategy with the above Ratio Pricing criterium to select the incoming column among those priced. If the minimum is not 0 STOP, the problem is infeasible.

Step 3: Use the Coupling Constraints Algorithm to obtain the optimum.

The difference with the Coupling Constraints Algorithm lies in Step 2, which can be implemented efficiently using the GBBF method. Hopefully when achieving feasibility in the common rows at the end of Step 2 the problem will be optimal or near optimal requiring only a few Phase 2 iterations in Step 3.

Implementation

Let $\Pi = (\Pi_0, \Pi_1, \ldots, \Pi_k)$ be the dual prices associated with the objective function for the current basis and $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_k)$ be the dual prices associated with the sum of infeasibilities for the current basis. Then when pricing out a non-basic column in block i, we need to compute

$$d_j = \sigma_0 D_j^i + \sigma_i G_j^i \qquad (4.10)$$

and

$$c_j = \Pi_0 D_j^i + \Pi_i G_j^i \qquad (4.11)$$

where D_j^i is the restriction of column j to common rows and G_j^i its restriction to rows in its own block i. Thus we only have to calculate σ_0, σ_i, Π_0 and Π_i and hence the number of operations in the two backward transformations will be proportional to two times the number of non-zeros in the representation of the Working Basis and in that of the inverse of block i. If the number of non-zeros in the representation of the Working Basis is less than in any one of the block inverses, then probably for problems with 3 or more blocks this would still be less than the number of operations in one backward transformation using a general representation for the problem inverse.

The pricing can be also done at only a small additional cost. For each non-basic column in block i compute first d_j by (4.10). If $d_j > -\delta$ then $j \notin SD_i$ and it is not necessary to compute c_j. Otherwise c_j is computed and we see which of the three rules applies.

Thus the pricing effort increases by the percentage of columns that have a negative reduced cost for the sum of infeasibilities. Especially in the later stages these can be expected to constitute a small fraction of the total number of non-basic columns.

4.1-6 Some Comments

Extensive experimental tests and comparisons in a systematic way are necessary to determine which of the above strategies will perform better for block-angular linear problems

with coupling constraints when using the GBBF Simplex Method. Up to now there are only a few experimental results (see [23], [38]) and no comparisons available, which stresses the need for Systems Optimization Laboratories (see Dantzig [6]) to remedy this situation .

From practical considerations primal strategies have the advantage (besides the savings in BTRAN when using GBBF) that they can be used in conjunction with almost all combinations of pricing and pivoting criteria used in current production codes, so for instance besides pricing out at each pricing operation only a subset of the non-basic variables (partial pricing), several candidates may be selected and updated (multiple pricing) using greatest change criteria in a sub-optimization involving only these candidates to determine which ones are to be introduced to the basis and in what order.

4.2. Block-Angular Linear Problems with Coupling Variables

This is the special case of the general block-angular problem P when there are no coupling constraints. It is the dual problem to the block-angular problem with coupling constraints analysed in 4.1. Of course one solution strategy could be to dualize and then use anyone of the methods in 4.1. It is however of interest to have methods that solve it directly, since this structure arises often in applications involving uncertainty (see [25]).

Also for block-angular linear problems with coupling variables there are some methods that have been around for a

long time and for which no experimental results are available.

4.2-1 Beale's Pseudobasic Variables Method [3]

Beale's method uses a primal strategy. The central idea is to treat the coupling variables as parameters in order to preserve a square block-angular structure for the basis. After a closer look it becomes apparent that they are only treated as parameters in the first part of the algorithm, where they are fixed at a value for which the system is assumed to have a solution. Later on they are really treated as basic variables; however the algebra of the Simplex Method has been modified so that the values of the incoming variable and of the other basic variables are expressed as a function of one of the coupling variables which is then allowed to change value until one of the basic variables leaves the basic set. The value of the "linking parameters" are modified in this process exactly as all the other basic variables. Using GBBF Beale's method is equivalent to the specialization of the General Algorithm to the case when there are no coupling constraints.

4.2-2 Gass' Dualplex Method [15]

As has been pointed out before, this method can be viewed as the dual of Rosen's algorithm (see 4.1-2). It assumes that for a given $Y = Y_0$ fixed all block problems have a feasible solution. Its strategy expressed in terms of the GBBF Simplex

Method is:

Step 1: Set k = 0. Optimize each block problem for $Y = Y_0$. If any problem is unbounded STOP, the whole problem is unbounded. Otherwise go to Step 2.

Step 2: Set k = k + 1. Solve a restricted problem where only coupling variables are allowed to enter the basic set. If an unbounded solution is encountered STOP, the problem is unbounded. Otherwise continue to Step 3.

Step 3: Let Y_k and Π_k be the values for the Y variables and for the dual prices in the optimal solution at the end of step 2. Fix Y_k as a parameter. For each block use Π^k to price out non-basic block variables. If all price out optimal STOP, the current solution is optimal. Otherwise exchange as many pseudo-basic variables as possible with non-basic variables that price out non-optimal. (Observe that when fixing Y^k as a parameter B_N becomes the true basis, with all previously pseudo-basic variables at value 0. Since we pivot only in rows of pseudo-basic variables the value of the solution does not change, and hence at the end the solution from Step 2 is still feasible, but not basic anymore). Return to Step 2.

4.2-3 Other Strategies

Both Beale and Gass assume the knowledge of an initial $Y = Y_0$ for which each block problem has a feasible solution. If this is not the case the problem can be set up in the usual way, a Phase 1 Procedure to minimize the sum of infeasibilities. If this minimum is greater than zero the problem is infeasible.

Otherwise the value of Y in the first feasible solution is used as Y_0 to start either method in Phase 2. With this added feature Beale's method is equivalent to the specialization of the General Algorithm in section 3.5-2 for the case when there are no coupling constraints.

Alternatively the PBP strategy with the ratio-pricing criterium to select the incoming variable could be used in Phase 1.

4.3. Block-Angular Linear Problems with Coupling Constraints and Variables

Only a few algorithms have been proposed in the literature for the general problem P. They usually were worked out as extensions of algorithms for the block-angular linear case with coupling constraints, like for instance the Generalized GUB method (see 4.1-1) and Rosen's algorithm (see 4.1-2). Extensions of the Dual and Primal Dual strategies have not been presented, probably because of the requirement of having a dual feasible solution at hand to start the procedures.

4.3-1 Primal Strategy: Hartman and Lasdon's Method [20]

Hartman and Lasdon use the usual Primal Simplex Strategy. They develop a basis factorization scheme in which a column corresponding to a block variable that becomes pseudo-basic is dropped from the block basis together with the row in which it was basic, thus reducing the size of the block basis. Hence all basis' may vary in size. Besides they do not require the Working

Basis to be minimal. To avoid it increasing too much, checks are done which require doing some computation.

They present some computational results of their method for a special class of production and inventory problems. For a given problem, the smaller the size of the Working Basis, the faster the iterations. In particular overall solution time is smallest if coupling columns are not introduced into the basis in Phase 1 (unless necessary), to keep the Working Basis small. These observations agree with our analysis in 3.5.

4.3-2 Ritter's Method [31]

Ritter's method amounts to a generalization of Rosen's method to problems having also coupling variables. It uses the same relaxation strategy as Rosen's, (see also Geoffrion [16] and 4.1-2) with a slightly different criteria as to which variables to relax and which violated relaxed variables to enforce to account for the presence of the coupling variables. All comments on Rosen's method (see 4.1-2) apply also here (with some slight modifications in some cases). We state his method under the assumption that a Y_0 is known for which all blocks have a feasible solution.

Step 1: For $Y = Y_0$ fixed, optimize all blocks.

Step 2: Relax non-negativity constraints on variables corresponding to the block basis'. Solve the relaxed problem (no restrictions on coupling variables). If the relaxed problem has no solution STOP, the problem is infeasible.

Step 3: Check if the relaxed non-negativity constraints are

satisfied. If so STOP, the solution is optimal. Otherwise rearrange variables between blocks and Working Basis as in Rosen's algorithm for at least one block having a variable not satisfying the non-negativity constraints. Whenever according to this rule a coupling variable associated with the Working Basis has to be switched with a block basic variable in its own block, the size of the Working Basis is instead increased by adding the block basic-column and its pivot row to it. This way at least one column associated with an infeasible variable is introduced to the Working Basis. Return to step 2.

Observe that when using the GBBF approach, besides step 1, also step 2 is the same as Rosen's and Ritter's algorithms, i.e. relax non-negativity constraints on basic variables not in the Working Basis.

Only the rearrangement procedure in step 3 is more general in Ritter's method to account for the coupling variables. Thus the five observations which we presented earlier in 4.1-2 after Rosen's method apply here.

4.3-3 Other Strategies

The General Algorithm presented in 3.5 is another example of a primal strategy, more refined than the usual simplex strategy to get the most out of the structure of the problem. The ratio-pricing technique described in 4.1-5 is another possibility that looks promising and can be extended directly to problems with coupling variables and constraints because it does not require dual feasibility. It was not incorporated into

the General Algorithm because it is still untested, as compared to the Partial Block Pricing strategy with the most negative reduced gradient criteria used in CCA. If tests later show that Partial Block Pricing with the ratio-pricing criterium is more efficient, it should be incorporated into the General Algorithm.

Of the strategies for block-angular linear problems with coupling constraints that require dual feasibility, the easiest to extend is Orchard-Hays parametric strategy (see 4.1-4). Recall that in this approach, after solving all block problems, the right hand side in the common rows is changed to force the current solution to be both dual and primal feasible. For problems with coupling constraints and variables the same thing can be done also to the cost coefficients in the coupling constraints to force them dual feasible after all block problems have been optimized. After this, both the modified cost coefficients on the coupling variables and the modified right hand side of the common rows are forced back to their old values using a parametric technique.

4.4. Specializations of the General Algorithm

To end this Chapter we want to mention the specializations of the General Algorithm for some special cases.

For block-angular linear problems with coupling constraints it reduces to the CCA method described in 3.4. Furthermore, if each block corresponds to a GUB set, then the CCA method in turn reduces to the GUB algorithm (see [12]), in which for each GUB set we initially select as key variable the one making the GUB

set feasible and giving the best value for the objective function.

For block-angular problems with coupling variables the General Algorithm specializes to the same strategy as Beale's Pseudo-basic Variables Method (see 4.2-1).