Mathematical Programming 31 (1985) 357-363
North-Holland

DEALING WITH DEGENERACY IN REDUCED GRADIENT ALGORITHMS

R.S. DEMBO*
School of Organization and Management, Yale University, New Haven, CT 06520, USA

and

J.G. KLINCEWICZ
Bell Laboratories, Holmdel, NJ 07733, USA

Received 11 July 1983
Revised manuscript received 13 September 1984

Many algorithms for linearly constrained optimization problems proceed by solving a sequence of subproblems. In these subproblems, the number of variables is implicitly reduced by using the linear constraints to express certain 'basic' variables in terms of other variables. Difficulties may arise, however, if degeneracy is present; that is, if one or more basic variables are at lower or upper bounds. In this situation, arbitrarily small movements along a feasible search direction in the reduced problem may result in infeasibilities for basic variables in the original problem. For such cases, the search direction is typically discarded, a new reduced problem is formed and a new search direction is computed. Such a process may be extremely costly, particularly in large-scale optimization where degeneracy is likely and good search directions can be expensive to compute. This paper is concerned with a practical method for ensuring that directions that are computed in the reduced space are actually feasible in the original problem. It is based on a generalization of the 'maximal basis' result first introduced by Dembo and Klincewicz for large nonlinear network optimization problems.

Key words: Degeneracy, Reduced Gradient Methods, Nonlinear Optimization.

1. Introduction

Many algorithms for linearly-constrained mathematical programming problems, such as the reduced gradient algorithm (see, for example [1, 4]), proceed by solving a sequence of subproblems in which the number of variables has been implicitly reduced. These reduced problems are obtained by using the linear constraints to express certain variables, designated as 'basic', in terms of other variables.

Within these subproblems, a new feasible solution is obtained from a previous one by moving along a computed 'search direction'. (The manner in which the search direction is calculated differs from algorithm to algorithm.) Difficulties may arise, however, if degeneracy is present, that is, if one or more basic variables are at lower or upper bounds. In this case, it is possible that an arbitrarily small move along the computed search direction in the reduced problem might violate the upper or lower bound of one or more basic variables in the original problem. If this occurs,

* Research supported in part by NSF Grant ECS-8119513 and DOT Research Grant CT-06-0011.

it may be necessary to form a new reduced problem and to then recalculate the search direction. Since in nonlinear optimization most of the computational effort may go into computing a search direction, the computation time spent in finding search directions that must be discarded can be significant. For small problems, this does not pose a difficulty because one can usually find a nondegenerate basis. This is not the case for larger problems where a nondegenerate basis might not exist.

The problem of degeneracy is particularly acute in large nonlinear network optimization problems. In this context, Dembo and Klincewicz [4] provided a mechanism to avoid computing useless search directions. The purpose of this paper is to extend the results of [4] to general linearly-constrained problems.

In Section 2, we introduce our notation and describe the problems of degeneracy more fully. Section 3 describes the concept of a 'maximal basis', which we propose as a remedy to the problems posed by degeneracy. The process of maintaining such a maximal basis is then outlined in Section 4. Finally, Section 5 offers some concluding remarks.

2. The problem

The linearly-constrained nonlinear programming problem can be stated as follows:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x) \tag{1}$$

$$\text{subject to} \quad Ax = b, \tag{2}$$

$$l \leqslant x \leqslant u, \tag{3}$$

where $f:\mathbb{R}^n \to \mathbb{R}$, A is an $m\times n$ matrix of rank m, and $u\in\mathbb{R}^n$ and $l\in\mathbb{R}^n$ are given upper and lower bounds respectively.

A common approach to solving such problems (see, for example, [1, 3, 4, 5, 8, 9]) is as follows. Let A be partitioned into basic, superbasic and nonbasic components (this terminology is due to Murtagh and Saunders). That is,

$$A = [B \;\; S \;\; N], \tag{4}$$

where B is an $m\times m$ nonsingular matrix (i.e., a basis for the constraints $Ax = b$). The superbasic columns S correspond to those variables outside the basis whose values are subject to change at a given iteration, and the nonbasic columns N correspond to variables whose values are held fixed. Using this partition, we also define and partition the following column vectors:

$x \equiv [x_B\, x_S\, x_N]^{\mathrm{T}}$ = vector of current values,

$g(x) \equiv [g_B\, g_S\, g_N]^{\mathrm{T}}$ = gradient of the objective function,

$p \equiv [p_B\, p_S\, p_N]^{\mathrm{T}}$ = current search direction, and similarly for vectors l and u. In this notation, (2) becomes

$$Bx_B + Sx_S + Nx_N = b. \tag{5}$$

Hence, the basic variables can be expressed as a function of the superbasic and nonbasic variables as follows:

$$x_B(x_S, x_N) = B^{-1}[b - Sx_S - Nx_N]. \tag{6}$$

Thus, the above problem (1), (2), (3) may be written equivalently as:

$$\underset{x_S, x_N}{\text{minimize}} \quad f(x_B(x_S, x_N), x_S, x_N) \tag{7}$$

$$\text{subject to} \quad l_B \leqslant x_B(x_S, x_N) \leqslant u_B, \tag{8}$$

$$l_S \leqslant x_S \leqslant u_S, \tag{9}$$

$$l_N \leqslant x_N \leqslant u_N. \tag{10}$$

If, in fact, the nonbasic variables x_N are held fixed and the basic variables expressed as in (6), then we obtain a restricted subproblem in which the objective function, derived from (7), can be expressed as $\hat{f}_S(x_S)$, a function that depends only upon the superbasic variables x_S. The chief idea behind this is that locally (in a neighborhood of the current point) the problem (1), (2), (3) reduces to the following problem with only simple bounding constraints:

$$\underset{x_S}{\text{minimize}} \quad \hat{f}_S(x_S) \tag{11}$$

$$\text{subject to} \quad l_S \leqslant x_S \leqslant u_S, \tag{12}$$

that is, provided none of the basic variables, x_B, is at its bound. We will refer to problem (11), (12) as the reduced problem.

When the basis is not degenerate, the problem (1), (2), (3), is termed '*locally equivalent*' to the reduced problem (11) and (12). By this we mean that there exists a neighborhood around the current point x_S such that for any point in the neighborhood, (8) remains satisfied. In this case, if we were to compute a feasible search direction for the superbasic variables and to move along this direction away from the current point, we would be assured of being able to take a nonzero step before encountering bounding constraints (8) or (9), provided x_S is not at a bound.

If a basic variable is at its bound, that is, if the basis is degenerate, one does not simplify the problem much by eliminating the equality constraints (2) and the basic variables x_B, since new binding inequality constraints (8) are created. In this case, it is possible to calculate a search direction that is feasible for the superbasic variables but for which an arbitrarily small movement would cause one or more bounding constraints (8) to be violated. Hence the reduced and original problems are no longer 'locally equivalent'. However, as we will show in Section 3, it is possible to construct a restriction of the reduced problem that is locally equivalent to the corresponding restriction of (1), (2).

3. Reducing the problem to that encountered in LP degeneracy

To resolve the difficulties alluded to above, we propose a generalization of the maximal basis concept first introduced by Dembo and Klincewicz [4] in the context of nonlinear network flow problems.

We first need some terminology. Let us call a variable 'free' if its current value is not at a bound. We will say that a column of A is free if it corresponds to a free variable and is not free otherwise.

Definition (*Maximal basis*). A basis is said to be maximal if there is no other basis with more free columns.

In the proposition below we see that finding a maximal basis is computationally inexpensive.

Proposition 1. *A maximal basis can always be formed using a greedy algorithm.*

To execute a greedy algorithm, first mark all the free columns of A. Try to include the free columns in the basis one at a time. If at any stage a free column is linearly dependent on the columns that are already in the basis, drop it from consideration. When no free columns remain, use remaining nonfree columns to complete the basis. The resulting basis is maximal.

Proof. Let r be the rank of the free columns and let g be the rank of a matrix M constructed by a greedy algorithm; if $g < r$, then there exists a free column rejected by the greedy algorithm that does not lie in the smaller column space spanned by M, contradicting the rejection rule. $\square$

In order to better appreciate the benefits of a basis that is maximal, we need to draw on a definition from the literature on matroids.

Definition (*Circuit*). Let B be a square m-dimensional matrix whose columns span $\mathbb{R}^m$ and let $y \in \mathbb{R}^m$ be some arbitrary vector. A *circuit* is then a *minimal* subset of the columns of B upon which y is linearly dependent.

Proposition 2. *Let B be a maximal basis and let S consist of columns corresponding only to free out-of-basis variables. Then the circuits induced by the columns of S are only made up of free columns.*

Proof. Any free column in S is, by construction, a linear combination of free basic columns and hence this property is inherited by the circuits formed by the columns of S. $\square$

Proposition 3. *If B is a maximal basis and S contains only free columns, then the reduced problem* (11), (12) *is locally equivalent to the original problem* (1), (2) *and* (3).

Proof. Since circuits induced by S contain only columns corresponding to free variables, there is a neighborhood in which arbitrary perturbations in x_S induce feasible perturbations in x_B. □

Remark. With a maximal basis and using only free superbasic variables one knows *a priori* that a nonzero movement along a search direction is possible. It is possible at nonoptimal points (such as at a vertex) for a situation to arise in which, even though the basis is maximal, it may be difficult to identify feasible descent directions for nonbasic variables that do not induce infeasibilities in basic variables. However, this is analogous to a degenerate vertex situation in linear programming which can be resolved in a finite number of pivots using a standard simplex pivoting rule such as in [2].

4. Maintaining a maximal basis

In a restriction strategy such as the one above, at any iteration there are three possibilities (assuming at the start of the iteration that the basis is maximal and the superbasic variables are all free):

(i) a basic variable hits a bound, or
(ii) a superbasic variable hits a bound, or
(iii) no variables move to a bound.

In cases (ii) and (iii) the basis remains maximal. If, in case (ii), the superbasic set is updated to exclude the variable that hit its bound, then the reduced problem is once again locally equivalent to the original one. If case (i) occurs, the basis may no longer be maximal. To restore maximality, we replace, if possible, basic variables that have hit bounds by superbasic variables that are free. (This might not be possible if the basic variable that hits its bound is in a circuit with a superbasic variable that simultaneously hits its bound.)

Identifying superbasic variables that are eligible to replace a given basic variable can be accomplished in a straightforward manner. For a given basic variable, the superbasics that are eligible to replace it are given by the nonzero elements in the corresponding row of $B^{-1}S$. Since most successful algorithms store B in factored form (e.g., $B = LU$ where L and U are lower and upper triangular matrices), finding the nonzero elements in a row of $B^{-1}S$ can be accomplished inexpensively. Specifically, let the jth basic variable be the one that is to be replaced and let e_j be the jth unit vector. The procedure is:

Solve $\quad B^{\mathrm{T}}y = e_j$;

Compute $\quad z = S^{\mathrm{T}}y$.

The nonzero elements of z then correspond to superbasic variables that are eligible to replace the jth basic variable.

5. Summary and conclusions

Consider two algorithms, identical in every respect except that one uses a maximal basis and the other does not. What are the relevant tradeoffs?

For the one that uses a maximal basis, some pivoting is needed occasionally to restore maximality of B and to resolve situations akin to LP degeneracy. However, at any iteration, if only free superbasic variables are moved, then one is assured of a nonzero movement along the search direction. When a nonbasic variable at a bound has to be moved, pivoting rules such as Bland's rule in [2] will ensure that a search direction is found in a finite number of steps. The algorithm that does not use a maximal basis must pivot when it is discovered that a given change in superbasic variables will result in an infeasible move for basic variables. In such cases, the work done to get a superbasic search direction is wasted.

Thus the tradeoff is perhaps an extra few pivots versus some wasted effort in computing a search direction. For algorithms that use expensive (e.g., modified Newton) directions, it seems that the tradeoff would be in favor of using a maximal basis.

There are a number of implementations of the maximal basis idea for the special case of large-scale nonlinear network flow problems [3, 4, 5]. These network algorithms include specialized versions of a scaled reduced gradient algorithm and a Newton algorithm. In the experiments reported on in [3, 4, 5] it has been found that the cost of maintaining a maximal basis is negligible compared to the cost of solving the problem.

Finally, it is natural to ask how this idea can be used in linear programming algorithms. It can have no effect on the simplex method since by definition all bases are maximal (all out-of-basis variables are at their bounds). However, it may make a difference for reduced gradient algorithms for linear programming [7]. This has yet to be explored.

Acknowledgment

We gratefully acknowledge Alan Hoffman's suggestion that we extend the results in [4]; that suggestion provided the motivation for this paper. We also would like to thank one of the referees for a careful reading which helped improve the exposition in this paper.

References

[1] J. Abadie and J. Carpentier, "Generalization of the Wolfe reduced gradient method to the case of nonlinear constraints", in: R. Fletcher, ed., *Optimization* (Academic Press, New York, 1969) pp. 37–47.

[2] R.G. Bland, "New finite pivoting rules for the simplex method", *Mathematics of Operations Research* 2 (1977) 103–107.

[3] R.S. Dembo, "A primal truncated-Newton algorithm with application to large-scale nonlinear network optimization", Working Paper No. 72, Series B, School of Organization and Management, Yale University (New Haven, CT, 1983).

[4] R.S. Dembo and J.G. Klincewicz, "A scaled reduced gradient algorithm for network flow problems with convex separable costs", *Mathematical Programming Study* 15 (1981) 125–147.

[5] J.G. Klincewicz, "A Newton method for convex separable network flow problems", *Networks* 13 (1983) 427–442.

[6] E.L. Lawler, *Combinatorial optimization, networks and matroids* (Holt, Rinehart & Winston, New York, 1976).

[7] C.E. Lemke, "The constrained gradient method of linear programming", *Journal of the SIAM* 9 (1961) 1–17.

[8] B.A. Murtagh and M.A. Saunders, "Large scale linearly constrained optimization", *Mathematical Programming* 14 (1978) 41–72.

[9] D.F. Shanno and R.E. Marsten, "Conjugate gradient methods for linearly constrained nonlinear programming", *Mathematical Programming Study* 16 (1983) 149–161.