A NEWTON-BASED ITERATIVE METHOD FOR FINDING A FEASIBLE POINT OF A SYSTEM OF CONVEX INEQUALITIES

Khaled S. Al-Sultan
Department of Industrial & Operations Engineering
The University of Michigan
Ann Arbor, MI 48109-2117
and
Department of Systems Engineering
King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia

Katta G. Murty and T. Yi
Department of Industrial & Operations Engineering
The University of Michigan
Ann Arbor, MI 48109-2117

Technical Report 91-17

May 1991

ABSTRACT

We present a surrogate constraint modification of Bregman's [1965] method of successive projection for finding a point in a convex set defined by a system of inequality constraints involving smooth convex functions. Preliminary computational experience with this algorithm is very encouraging.

1 Introduction

Let $\mathbf{K}$ be the set of feasible solutions of the system of constraints

$$g_i(x) \leq 0 \ , \quad i = 1 \text{ to } m \tag{1}$$

where each $g_i(x)$ is a smooth convex function defined over $\mathbf{R}^n$. Bregman [1965] proposed the following iterative algorithm for computing a point in $\mathbf{K}$.

Initialization: Initiate some point $x^0 \in \mathbf{R}^n$.

Main Step: Let x^r be the current point. If x^r satisfies (1), $x^r \in \mathbf{K}$, terminates. Otherwise, select a constraint in (1) violated by x^r. Suppose it is the ith. So $g_i(x^r) > 0$. Define x^{r+1} = nearest point by Euclidean distance in the set $\{x : g_i(x) \leq 0\}$. Compute x^{r+1} and go to the next step.

Under some mild conditions, Bregman [1965] proved that the sequence $\{x^0, x^1, \cdots\}$ converges to a point in $\mathbf{K}$ if $\mathbf{K} \neq \emptyset$.

Recently, Yang and Murty [1990] have investigated a similar algorithm on large systems of linear inequalities. They found that the convergence is speeded up substantially by projecting on a surrogate constraint which is a positive linear combination of all the violated constraints, in each step. In this paper, we discuss the same surrogate constraint version of the algorithm for solving (1) here, and report some preliminary computational results with it.

2 The Algorithm

Here is the statement of the surrogate constraint modification of the method of successive projections. We will call it the *surrogate constraint method.*

Initialization: Initiate with some point $x^0 \in \mathbf{R}^n$.

Main Step: Let x^r be the current point. Compute $I(x^r) = \{i : g_i(x^r) > 0\}$. If $I(x^r) = \emptyset$, $x^r \in \mathbf{K}$, terminate. If $I(x^r) \neq \emptyset$, select positive weight vector $\Pi^r = (\Pi^r : i \in I(x^r))$, and generate the surrogate constraint function $g(x) = \sum_{i \in I(X^r)} \Pi_i^r g_i(x)$. Now, solve the nearest point problem

$$\begin{aligned} \text{minimize} \quad & \|y - x^r\| \\ \text{subject to} \quad & g(y) \leq 0 \end{aligned} \tag{2}$$

where $\|\cdot\|$ is the Euclidean norm. Let x^{r+1} be an optimum solution for (2). With x^{r+1}, go to the next step.

3 How To Solve (2)

(2) is the problem of finding the nearest point by Euclidean distance to x^r in a convex set. The set of feasible solutions of (2) includes $\mathbf{K}$. So, under the assumption that $\mathbf{K} \neq \emptyset$, the set of feasible solutions of (2) is nonempty, and hence it has an optimum solution and the constraint in (2) will hold as an equation at the optimum solution for it. From this we see that the KKT conditions for (2) are

$$\begin{aligned} 2(y - x^r) + \lambda \nabla g(y) = 0 \ . \\ g(y) = 0 \end{aligned} \tag{3}$$

where $\nabla g(y)$ is the gradient vector of $g(y)$ written as a column vector, and $\lambda \geq 0$ is the Lagrange multiplier. (3) is a system of $(n+1)$ equations in $(n+1)$ unknowns, we apply Newton's method to solve it, leading to the following iterative scheme.

Initialization: Initiate with $y^0 = x^r$ and a $\lambda_0 > 0$.

Main Step: Let y^r be the current point. Solve the following system of linear equations for (ζ, δ).

$$\begin{bmatrix} 2I + \lambda \mathbf{H}(g(y^r)) & \nabla(y^r) \\ (\nabla g(y^r))^T & 0 \end{bmatrix} \begin{bmatrix} \zeta \\ \delta \end{bmatrix} = - \begin{bmatrix} 2(y^r - x^r) + \lambda_s \nabla g(y^r) \\ g(y_1^r) \end{bmatrix} \tag{4}$$

where $\mathbf{H}(g(y^r))$ is the hessian matrix of $g(x)$ evaluated at $x = y^r$, and I is the unit matrix of order n. Let $y^{r+1} = y^r + \zeta$, $\lambda_{s+1} = \lambda_s + \delta$. If $\lambda_{s+1} > 0$ and (y^{s+1}, y^{s+1}) satisfies (3), then y^{s+1} is an optimum solution of (2), terminate this scheme and define $\lambda^{r+1} = y^{s+1}$. If (y^{s+1}, λ_{s+1}) does not satisfy (3), then go to the next step in the scheme with them.

Under mild conditions it can be shown that (4) always has a solution (ζ, δ), that λ remains positive in this scheme, and that the sequence of points generated in the scheme $\{(y^r, \lambda_r) : s = 0, 1, \cdots\}$ converges to a solution of (3); and hence y^r converges to the optimum solution of (2).

4 Choice of Weights for Generating the Surrogate Constraint

A variety of strategies can be used for selecting the weight vector $\Pi^r = (\Pi_i^r : i \in I(x^r))$. One strategy selects all the weights to be equal, i.e., $\Pi_i^r = \frac{1}{|I(x^r)|}$, for all $i \in I(x^r)$. Another strategy selects $\Pi_i^r = \frac{g_i(x^r)}{(\sum(g_i(x^r) \text{: over } i \in I(x^r)))}$. This latter strategy makes the weight for a violated constraint proportional to the extent of violation measured by the function value. A third strategy takes the weight-vector to be a convex combination of the two strategies mentioned above. These are the weight vectors investigated by Yang and Murty [] in their work with systems of linear inequalities.

5 Generation of the Surrogate Constraint

An expensive piece of work in each step is that of finding all the constraints in (1) violated by the current point. This work is highly amenable to massively parallel implementation. We can use up to m separate processors operating in parallel, each one dedicated to checking a separate constraint or a small group of constraints in (1). This is a major advantage of this surrogate constraint method.

6 Convergence Results

We assume that $\mathbf{K} \neq \emptyset$.

Figure 1:

(a) Let $\{x^r : r = 0, 1, \cdots\}$ be the sequence of points generated by the surrogate constraint method, and let $\bar{x}$ be any point in **K**. Then

$$\|x^{r+1} - \bar{x}\| < \|x^r - \bar{x}\| \ . \tag{5}$$

See Figure 1. x^{r+1} is the nearest point in $= \{x : g(x) \leq 0\}$ to x^r, where $g(x)$ is the surrogate constraint constructed in the step when x^r is the current point.

So, by nearest point theory, there is a supporting hyperplane **H** to at x^{r+1} which is orthogonal to the line segment joining x^r and x^{r+1}. **K** $\in$, so, $\bar{x} \in$. Hence, the triangle determined by the points x^r, x^{r+1} and $\bar{x}$ is an obtuse angled triangle with its angle at x^{r+1} obtuse. Hence, the side of this triangle joining the points x^r and $\bar{x}$ is its longest side, i.e. (5) holds.

(b) Select any point $\bar{x} \in \mathbf{K}$. Let $S = \{x : ||x - \bar{x}|| \leq ||x^0 - \bar{x}||\}$ where x is the point with which the surrogate constraint algorithm is initiated. S is a compact subset of R^n, and by the result in (a), all the points in the sequence generated by the algorithm are contained in S. The algorithmic map is clearly closed. And the algorithm only terminates when the current point is in **K**. All these properties together imply that the sequence $\{x^r : r = 0, 1, \cdots\}$ converges to a point in **K** by the global convergence theorem of iterative descent algorithms [Bazaraa and Shetty 1979].

7 Computational Experience

The above algorithm was coded in FORTRAN 77 and tested on APOLLO 4000 machines. Due to the lack of standard test problems, we have selected problems from two sources.

TP(1) Test examples for nonlinear programming codes by W. Hock and K. Schittkowski, where we put the objective function $f(x)$ as a constraint or $f(x) \leq f(x^*)$ where x^* is the best solution they have found. This gives a test problem of the type that our algorithm can be applied to. We have tried problems 12, 19, 29, and 34 of which only problem 12 is out to be a system of convex inequalities while others are transformed to a system of nonlinear inequalities.

TP(2) Problems #1 and #2 of R. Schnable's systems of nonlinear inequalities.

We have tested with several strategies, among them

(1) Perform a single Newton step of the algorithm

This means that we do not solve the nearest point problem completely but rather solve only a system of linear equations in each iteration. Each iteration of this modified algorithm, of course, requires much less amount of work than an iteration of the original algorithm.

Since the constraint and objective functions are both convex, this problem has unique local minimum. Thus the KKT point with positive Lagrangian multiplier is the global optimum of the problem (1). The following is the KKT condition for this problem. Since we know the nearest point is always on the boundary, we can think of the constraint in (1) as an equality constraint.

(2) Pushing iteration points into the interior of the feasible region

This algorithm suffers most severely from the oscillation between two or more constraints. Figure 1 is a 2-dimensional example of this phenomena. To avoid this, we put the index into the active constraint set $I(x^k)$ when $g_i(x^k) = 0$, that is, x^k is on the boundary of the set $\{x : g_i(x) \leq 0\}$. Practically, the feasibility test in the main algorithm checks whether $g_i(x)$ is greater than a positive small number ϵ. Thus if $0 < g_i(x) < \epsilon$, then we can conclude x^k is on the boundary. Figure 2 shows the effect of this modification to the problem in Figure 1. The improvement after this modification has been roughly 40-50% in CPU time.

The complete results are shown for test problems TP1, and TP2 in Appendices 1 and 2. But we summarize some of our findings below:

Figure 2: Oscillation between two constraints.

Figure 3: The Effect of Pushing Iteration Point into Interior.

(1) The single Newton step version of the algorithm always converges and is always superior to the complete Newton versionerm of speed of convergence.

(2) The algorithm always converges for systems of nonlinear inequalities.

(3) As can be seen from figures in the appendices, feasibility (which we define as $\sum_{i=1}^{m} |g_i(x^k)|$) gets reduced exponentially at each iteration.

(4) It is clear that strategy II for selecting weights is always better than strategy I. For exmaple, see the results for problem 12.

Set

$$\pi_i = \frac{1}{\| \nabla g_i(x^k) \|}$$

Then, from $g(x^k) = \sum \pi_i g_i(x^k)$, we get

$$\begin{aligned} \nabla g(x^k) &= \sum \pi_i \nabla g_i(x^k) \\ &= \sum_{i \in I} \frac{\nabla g_i(x^k)}{\|\nabla g_i(x^k)\|} \end{aligned}$$

We will call this strategy II.

We have tested with both strategies I and II. Strategy II seems to be always superior to strategy I, and in some cases strate II converged only 20-40 steps while strategy I took more than 200 steps.

8 Computational Results

We have tried 8 sample problems. Three of them have convex constraints and the others have nonconvex constraints. In all cases both 1-step and full-step

Table 1. Computational Result for Some Sample Problems.

Problem			# of	# of	CPU Times(sec.)			
No.	Source	Con	Vars	Const	I	II	III	IV
1		C	2	3	0.21(4)	+	+	+
2		C	2	3	3.05(*)	2.00(*)	1.18(93)	+
3		NC	4	12	3.0(*)	1.10(44)	0.94(41)	+
4	Schnabel #1	NC	2	2	+	+	+	0.22(5)
5	Schnabel #2	NC	5	5	+	+	+	0.25(6)
6	Schittkowski #12	C	2	2	1.4(*)	1.31(*)	+	0.28(20)
7	Schittkowski #19	C	2	7	+	1.93(*)	+	0.36(13)
8	Schittkowski #29	C	3	2	+	+	+	0.38(33)
9	Schittkowski #34	C	3	9	2.74(*)	2.19(*)	+	2.33(*)

+ not tried, C convex, NC nonconvex

The number in the parentheses are the total number of main iterations. Asterisk(*) denotes that it took more than 200 iterations to attain required accuracy. The number in the parentheses are the total numbers of main iterations.

I, II, III, and IV under the CPU time column denotes the following configuration, respectively.

I: Full-step Newton, No pushing into interior points, π determining strategy I.

II: One-step Newton, No pushing into interior points, π determining strategy I.

III: One-step Newton, Pushing into interior points, π determining strategy I.

IV: One-step Newton, Pushing into interior points, π determingin strategy II.

methods converged when we choose a starting point which is not far from the feasible region. But the computational efficiency was much improved when the 1-step method was used. Also it is worth noting that the 1-step method is more robust. Even in the case that the full-step Newston method diverges, the 1-step Newton method sometimes converged to a feasible point. (This happens when the constraints are nonconvex.)

Also the strategies discussed in Section 4 improved the performance of the algorithm. The results are summarized in Table 1.

9 Conclusion

In this paper, an algorithm for finding a feasible point for a system of convex inequalities has been developed. Preliminary computational results are very encouraging, and although theoretical convergence is yet to be established for the case where the inequalities are nonconvex, computational experiments show that in all nonconvex problems the algorithm has always converged.

References

[1] M. Bazaraa, and C.M. Shetty, *Nonlinear Programming: Theory and Algorithms*, John Wiley, 1979.

[2] L.M. Bregman, "The Method of Successive Projection for Finding a Common Point of Convex Sets," *Soviet Mathematics Doklady*, Tom 162, No. 3, 1965.

[3] K.G. Murty, *Linear Complementarity, Linear and Nonlinear Programming*, Heldermann Verlag, 1988.

[4] R.B. Schnabel, "Defining Feasibility of a Set of Nonlinear Inequality Constraints," *Mathematical Programming Study 16*, March 1982.

[5] K. Yang, and K.G. Murty, "New Iterative Methods for Linear Inequalities," forthcoming in *JOTA*.