JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 62, No. 2, AUGUST 1989

An Extension of the Simplex Method to Constrained Nonlinear Optimization

M. B. SUBRAHMANYAM[1]

Communicated by F. Zirilli

Abstract. The simplex algorithm of Nelder and Mead is extended to handle nonlinear optimization problems with constraints. To prevent the simplex from collapsing into a subspace near the constraints, a delayed reflection is introduced for those points moving into the infeasible region. Numerical experience indicates that the proposed algorithm yields good results in the presence of both inequality and equality constraints, even when the constraint region is narrow. We note that it may be possible to modify and improve the algorithm by trying out variants.

Key Words. Function minimization, constrained optimization, Nelder-Mead simplex algorithm, nonlinear programming algorithms.

1. Introduction

The simplex algorithm was originally proposed by Spendley, Hext, and Himsworth (Ref. 1). The algorithm makes use of reflection of the worst vertex to arrive at a minimum. Nelder and Mead (Ref. 2) improved the performance of the algorithm by incorporating expansion, contraction, and shrinkage of the simplex. There have been some modifications of the algorithm by experimenting with different parameter values and by allowing unlimited expansion followed by translation of the simplex (Ref. 3).

The Nelder-Mead simplex algorithm has proved to be quite useful in the case of unconstrained optimization problems because of its robust properties. Studies have shown the algorithm to be very reliable in the case of minimization of noisy functions. Also, no calculation of the gradient is needed for the implementation of the algorithm. Although the algorithm is

[1] Electronics Engineer, Flight Control, Naval Air Development Center, Warminster, Pennsylvania.

0022-3239/89/0800-0311$06.00/0

widely used, theoretical convergence results are almost nonexistent. Woods (Ref. 4) gives a convergence theorem, along with a good description of the algorithm and a stopping criterion.

In this paper, we present an extension of the simplex algorithm to handle inequality constraints. Equality constraints can be incorporated by converting them to inequality constraints. The main additional operation that we use on the vertices of the simplex is what we call a delayed reflection, which will be described in the next two sections.

2. Problems with Constraints

The problem is to minimize a nonlinear, real-valued function $f(x_1, x_2, \ldots, x_n)$ subject to the inequality constraints

$$g_i(x_1, x_2, \ldots, x_n) \leq 0, \qquad i = 1, 2, \ldots, m.$$

In the case of unconstrained problems, the Nelder–Mead algorithm starts with an initial simplex and proceeds by moving the simplex away from the current worst vertex. The four operations for moving the simplex are reflection, expansion, contraction, and shrinkage. The original method of Spendley, Hext, and Himsworth (Ref. 1) was just to replace the worst vertex in the simplex by its reflection in the centroid of the others, thereby producing a new simplex on which the procedure can be repeated.

We order the vertices as $\langle v_1, v_2, \ldots, v_{n+1} \rangle$, such that v_{n+1} is the worst vertex. Let $\bar{v}$ be the centroid of $v_1, \ldots, v_n$. A reflected vertex is computed as

$$v_r = (1+\alpha)\bar{v} - \alpha v_{n+1},$$

where $\alpha > 0$ is the reflection coefficient.

There have been attempts to deal with constraints in the case of the simplex algorithm using penalty functions. Both Spendley, Hext, and Himsworth and Nelder and Mead proposed assigning large positive value if a vertex in the simplex becomes infeasible. Thus, the infeasible vertex becomes the worst vertex and will be reflected in the next iteration.

Box (Ref. 5) found that this approach mostly led to the simplex flattening itself against the constraints and collapsing into a subspace. Thus, the diameter of the simplex quickly goes to zero near constraints, and the result is convergence to a false minimum. To overcome this problem, Box developed the complex method where the complex has more than $n+1$ vertices. The complex method assumes that the feasible region is convex (see, for example, Ref. 6, p. 26). Our algorithm has been tested on problems with nonconvex feasible regions and was found to be successful.

The main reason for the ineffectiveness of the penalty function approach is that the infeasible vertex which now has the assigned high value is reflected toward the worst vertex of the previous iteration. This results in successively smaller simplices, and the simplices collapse into a subspace.

To prevent this collapse, we use a delayed reflection. If a vertex becomes infeasible, we do not increase the value at this vertex until the next iteration is completed. Thus, the next iteration is accomplished using the actual value of the function at the infeasible point. At the end of the iteration, in case the previous infeasible vertex is not the worst vertex, it is assigned a high value, so that it then becomes a candidate for reflection during the next iteration.

3. Description of the Algorithm

For the unconstrained case, we use the algorithm described in Ref. 4 with $\alpha = 0.95$, $\beta = 0.5$, and $\gamma = 2$. Here, α is the reflection coefficient, β is the contraction coefficient, and γ is the expansion coefficient. For constrained problems, the algorithm is described below.

Choice of the initial simplex. A feasible point $x = (x_1, x_2, \ldots, x_n)$ is selected. Let $\delta = 0.5$. Let e_i be the unit vector along the x_i-axis. The vertices $v_2, \ldots, v_{n+1}$ are selected as

$$v_i = x + e_{i-1}\delta x_{i-1}, \qquad i = 2, \ldots, n+1.$$

In case $x_{i-1} = 0$,

$$v_i = x + e_{i-1}\delta.$$

If any vertex becomes infeasible, repeat the above step by halving δ each time until the vertex becomes feasible.

The vertex v_1 is selected as

$$v_1 = (1 - \delta)x.$$

In case any component of x is zero, the corresponding component of v_1 is selected as $-\delta$. If v_1 is infeasible, the above step is repeated by halving δ each successive time until v_1 becomes feasible. Thus, we have an initial simplex with dimension n which is completely inside the feasible region.

The simplex algorithm which uses delayed reflection is described below.

Simplex Algorithm for Constrained Minimization. Given the initial simplex S_1 with ordered vertices $\langle v_1, v_2, \ldots, v_{n+1}\rangle$, set $\alpha = 0.95$, $\beta = 0.5$, and $\gamma = 2$. Thus, v_1 is the vertex with the least function value and v_{n+1} is the vertex with the greatest function value.

For the first iteration, we follow the same algorithm listed in Ref. 4 with some modifications, which is given below.

For $k=1$, compute

$$\bar{v}=(1/n)\sum_{i=1}^{n} v_i.$$

Compute

$$v_r=(1+\alpha)\bar{v}-\alpha v_{n+1}.$$

Set $v^k=v_r$. If $f(v_r)<f(v_n)$ and if $f(v_r)<f(v_1)$, then compute

$$v_e=\gamma v_r+(1-\gamma)\bar{v}.$$

If v_e is infeasible, then set $v^k=v_r$. If v_e is feasible and $f(v_e)<f(v_1)$, then set $v^k=v_e$.

If $f(v_r)\geq f(v_n)$, compute v^k in the following manner. If v_r is infeasible, then set $v^k=v_r$. If v_r is feasible, then set $v^t=v_{n+1}$. If v_r is feasible and $f(v_r)<f(v_{n+1})$, set $v^t=v_r$. Compute

$$v_c=\beta v^t+(1-\beta)\bar{v}.$$

If $f(v_c)<f(v_{n+1})$, then set $v^k=v_c$. If $f(v_c)\geq f(v_{n+1})$, then let

$$v_j=(v_1+v_j)/2, \text{ for } j=2,\ldots,n,$$

and

$$v^k=(v_1+v_{n+1})/2.$$

Sort $\langle v_1, v_2, \ldots, v_n, v^k\rangle$ to obtain

$$S_{k+1}=\langle v_1, v_2, \ldots, v_{n+1}\rangle.$$

For $k=2, 3, \ldots$, we follow the procedure given below. The last step is sorting

$$S_k=\langle v_1, v_2, \ldots, v_n, v^{k-1}\rangle$$

to obtain

$$S_k=\langle v_1, v_2, \ldots, v_{n+1}\rangle.$$

It is to be noted below that we assign a large positive value to the infeasible vertex only after sorting. This makes it a candidate for reflection during the next iteration.

If v^{k-1} is infeasible and $v_i=v^{k-1}$, set $f(v_i)=1E+100$. Compute

$$\bar{v}=(1/n)\sum_{i=1}^{n} v_i.$$

Also, compute

$$v_r = (1+\alpha)\bar{v} - \alpha v_{n+1},$$

and set $v^k = v_r$. If $f(v_r) < f(v_n)$ and if $f(v_r) < f(v_1)$, then compute

$$v_e = \gamma v_r + (1-\gamma)\bar{v}.$$

If v_e is infeasible, then set $v^k = v_r$. If v_e is feasible and $f(v_e) < f(v_1)$, then set $v^k = v_e$.

If $f(v_r) \geq f(v_n)$, then do the following. If v_r is infeasible or $f(v_{n+1}) > 1E+99$, set $v^k = v_r$. If both v_r and v_{n+1} are feasible, then set $v' = v_{n+1}$. If $f(v_r) < f(v_{n+1})$, then set $v' = v_r$, and compute

$$v_c = \beta v' + (1-\beta)\bar{v}.$$

If $f(v_c) < f(v_{n+1})$, then set $v^k = v_c$. If $f(v_c) \geq f(v_{n+1})$, then let

$$v_j = (v_1 + v_j)/2, \qquad j = 2, \ldots, n,$$

and

$$v^k = (v_1 + v_{n+1})/2.$$

Sort $\langle v_1, \ldots, v_n, v^k \rangle$ to obtain

$$S_{k+1} = \langle v_1, \ldots, v_{n+1} \rangle.$$

The stopping criterion used was to compute $\sum_{j=1}^{n+1} \|v_j - v_1\|_1$ and require it to be less than a prespecified tolerance. The norm $\|\cdot\|_1$ refers to the l_1-norm, which is the sum of the absolute values of the components of the vector.

The algorithm basically consists of reflection and expansion near the boundary of a constraint region, since it usually happens there that either the reflected point v_r is infeasible or the point being reflected v_{n+1} is infeasible. The convergence is obtained, since $\alpha = 0.95 < 1$. Smaller values of α yielded slightly inaccurate minima.

Variants of the algorithm were tried incorporating contraction and shrinkage near the boundary of the constraint region. The author was not able to better the algorithm using these variants. However, there always exists the possibility that better variants of the present algorithm can be found with detailed experimentation.

4. Numerical Tests

The algorithm was implemented in FORTRAN in double-precision on a Zentih Data Systems Z-248 PC series computer. The Z-248 PC is a 16-bit

machine. The compiler used was the Microsoft FORTRAN compiler version 3.20. Numerical tests were performed on problems up to seven variables. Since one of the advantages of the technique is robustness in the presence of noisy functions, it is felt that the method is of interest in itself and no attempt was made to compare the results with other constrained methods. Several examples are given below. Note that, in the results, the number of function evaluations refers to the number of evaluations of the objective function only.

Problem 4.1. Nonconvex, Two-Variable Example. The problem is (Ref. 7)

$$\min f(x) = (x_1 - 10)^3 + (x_2 - 20)^3,$$

subject to

$$g_1(x_1, x_2) = -x_1 + 13 \leq 0,$$

$$g_2(x_1, x_2) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0,$$

$$g_3(x_1, x_2) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0,$$

$$g_4(x_1, x_2) = -x_2 \leq 0.$$

The objective function is not convex, and the constraint region is a narrow winding (half-moon shaped) valley. The results for the initial point (14.35, 8.6) and a tolerance of $1E$-8 are given below:

$$x^* = (14.0950013, 0.8429636),$$

$$f(x^*) = -6961.8106875,$$

number of iterations = 809,

number of function evaluations = 833.

The solution agrees well with the one given in Ref. 7.

Problem 4.2. Rosen-Suzuki Test Problem. The problem, a convex programming problem, is (Ref. 7)

$$\min f(x) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4,$$

subject to

$$g_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - x_4 - 8 \leq 0,$$

$$g_2(x) = x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 - 10 \leq 0,$$

$$g_3(x) = 2x_1^2 + x_2^2 + x_3^2 + 2x_1 - x_2 - x_4 - 5 \leq 0.$$

The initial point is (0, 0, 0, 0) and the tolerance is $1E-4$. The exact solution is (0, 1, 2, −1) with $f_{\min} = -44$.

The solution with our algorithm is given below:

$$x^* = (1.9357E-5, 0.999971, 1.999994, -1.000012),$$

$$f(x^*) = -43.999999988,$$

number of iterations = 1390,

number of function evaluations = 1415.

Problem 4.3. Four-Variable Problem with an Equality Constraint. The problem is (Ref. 8)

$$\min f(x) = 2 - x_1 x_2 x_3,$$

subject to

$$x_1 + 2x_2 + 2x_3 - x_4 = 0,$$

$$x_i \geq 0, \qquad i = 1, 2, 3, 4,$$

$$1 - x_i \geq 0, \qquad i = 1, 2, 3,$$

$$2 - x_4 \geq 0.$$

We replaced the equality constraint by the inequality constraint

$$1E\text{-}8 - |x_1 + 2x_2 + 2x_3 - x_4| \geq 0.$$

The optimal solution is given by (2/3, 1/3, 1/3, 2), with the optimum value being 1.9259259.

With our algorithm, we obtained the following result with the initial point (0.1, 0.1, 0.1, 0.5) and a tolerance of $1E$-8:

$$x^* = (0.666666669, 0.3333333304, 0.3333333398, 2.0),$$

$$f(x^*) = 1.9259259,$$

number of iterations = 6322,

number of function evaluations = 7106.

Problem 4.4. Five-Variable Problem. The five-variable problem is problem 12.27 in Ref. 9. It has 15 inequality constraints; due to its length, it will not be repeated here. The solution is given in Ref. 9 as

$$x = = (4.5375, 2.3999, 60.0, 9.2999, 6.9999),$$

with

$$f(x) = -5.2802E+6.$$

The starting point is (2.52, 2.0, 37.5, 9.25, 6.8), and the tolerance is selected as $1E$-4.

With our algorithm, we arrived at the following solution:

$$x^* = (4.537356, 2.399963, 60.00023, 9.2999985, 6.9999995),$$

$$f(x^*) = -5280216.347,$$

number of iterations = 5098,

number of function evaluations = 5395.

Problem 4.5. Seven-Variable Problem. The seven-variable problem is taken from Ref. 10 and is to minimize

$$f(x) = (x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2 + 10x_5^6+7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7,$$

subject to

$$2x_1^2+3x_2^4+x_3+4x_4^2+5x_5-127\leq 0,$$

$$7x_1+3x_2+10x_3^2+x_4-x_5-282\leq 0,$$

$$23x_1+x_2^2+6x_6^2-8x_7-196\leq 0,$$

$$4x_1^2+x_2^2-3x_1x_2+2x_3^2+5x_6-11x_7\leq 0.$$

The solution given in Ref. 10 is

$$x = (2.33050, 1.95137, -0.47754, 4.36573, -0.62448, 1.03813, 1.59423),$$

with

$$f(x) = 680.630.$$

Our algorithm gave the following values with the initial point (1, 2, 0, 4, 0, 1, 1) and a tolerance of $1E-4$:

$$x^* = (2.330517, 1.9513717, -0.47759196, 4.365728, -0.6245119, 1.0381598, 1.5942702),$$

$$f(x^*) = 680.63005743,$$

number of iterations = 3933,

number of function evaluations = 4368.

5. Conclusions

The algorithm yielded accurate results for problems involving both inequality and equality constraints even when the constraint region is narrow. The new operation of delayed reflection prevented the collapse of the simplex near the boundary of the constraint region. Although the number of function evaluations needed is large, it is hoped that further work will result in a more efficient variant of the algorithm.

References

1. SPENDLEY, W., HEXT, G. R., and HIMSWORTH, F. R., *Sequential Application of the Simplex Design in Optimization and Evolutionary Operation*, Technometrics, Vol. 4, pp. 441–461, 1962.
2. NELDER, J. A., and MEAD, R., *A Simplex Method for Function Minimization*, Computer Journal, Vol. 7, pp. 308–313, 1965.
3. PARKINSON, J. M., and HUTCHINSON, D., *An Investigation into the Variants of the Simplex Method*, Numerical Methods for Nonlinear Optimization, Edited by F. A. Lootsma, Academic Press, London, England, pp. 115–135, 1972.
4. WOODS, D. J., *An Interactive Approach to Solving Multi-Objective Optimization Problems*, Technical Report No. 85-5, Department of Mathematical Sciences, Rice University, Houston, Texas, 1985.
5. BOX, M. J., *A New Method of Constrained Optimization and a Comparision with Other Methods*, Computer Journal, Vol. 8, pp. 42–52, 1965.
6. KOWALIK, J., and OSBORNE, M. R., *Methods for Unconstrained Optimization Problems*, American Elsevier, New York, New York, 1968.
7. GOULD, F. J., *Nonlinear Tolerance Programming*, Numerical Methods for Nonlinear Optimization, Edited by F. A. Lootsma, Academic Press, London, England, pp. 349–366, 1972.
8. MIELE, A., CRAGG, E. E., and LEVY, A. V., *Use of the Augmented Penalty Function in Mathematical Programming Problems, Part 2*, Journal of Optimization Theory and Applications, Vol. 8, pp. 131–153, 1971.
9. BETTS, J. T., *An Accelerated Multiplier Method for Nonlinear Programming*, Journal of Optimization Theory and Applications, Vol. 21, pp. 137–174, 1977.
10. DI PILLO, G. and GRIPPO, L., *A New Augmented Lagrangian Function for Inequality Constraints in Nonlinear Programming Problems*, Journal of Optimization Theory and Applications, Vol. 36, pp. 495–519, 1982.