LIBERTAS MATHEMATICA , VOL. 8 (1988)

PARALLEL LINE SEARCH ALGORITHMS FOR SOLVING CONVEX UNCONSTRAINED MINIMIZATION PROBLEMS

Prapasri Asawakun and Florian A. Potra

Abstract. In this paper, we consider the problem of computing the minimum of a function, $f : \mathbb{R}^n \to \mathbb{R}$, where f is twice continuously differentiable and convex. Our approach is to search on a preassigned set of linearly independent directions, and our aim is to examine and find a possibility of developing a parallel algorithm to solve this unconstrained minimization problem, as we are motivated by the availability of parallel computers. The parallelism in our algorithm will be confined to doing line search in parallel along mutually conjugate directions, which, in our implementation, are generated by employiung the update formula to the approximation of the inverse Hessian.

1. Introduction.

Parallel algorithms for solving an unconstrained minimization problem have been investigated even before the availability of parallel computers. Some parallel features are based on simultaneous evaluations of the objective function at different points or evaluations of the estimated gradients and Hessians. Therefore, these can be introduced into the already existing sequential algorithms. For a survey of parallel nonlinear optimization see Lootsma and Ragsdell (1988). We shall describe briefly the methods of Chazan and Miranker (1970), and Sutti (1983) which are mentioned in this survey and which are somewhat related to our approach.

Chazan and Miranker (1970) give an algorithm for unconstrained minimization of

strictly convex functions which is based on Powell's minimization procedure without calculating derivatives. Firstly, a set U of n linearly independent n–vectors $u_1,u_2,\ldots,u_n$ is selected. Given a starting point x^0 and $n-1$ vectors $v^1,\ldots,v^{n-1}$, at each iteration $k = 1,2,3,\ldots$, we select a vector u_j, $j \equiv k \pmod n$ from U and call this v^n. Now, let $x^i = x^{i+1}+v^i$, $i = 1,\ldots,n$ (a vertex along a polygonal line). From these $x^1,x^2,\ldots,x^n$, we perform n line searches along a common direction v^1 and obtain the steplengths $\alpha_1,\ldots,\alpha_n$ along v^1 respectively. Before the beginning of the new iteration, we update x^0 and $v^1,\ldots,v^{n-1}$ by taking x^0 to be $x^1+\alpha_1 v^1$ and v^i to be $v^{i+1}+(\alpha_{i+1}-\alpha_i)v^1$, $i = 1,\ldots,n-1$. This completes one iteration. This method gives the exact solution for a quadratic objective function in at most n iterations. Their proposed parallelism is therefore to perform geometrically parallel line search in parallel.

Sutti (1983) also modifies Powell's method (nongradient method) by taking a complete set of n linearly independent normalized vectors $d_1,\ldots,d_n$ and performing line searches from a current estimate x^i along these directions in parallel. Sutti distinguishes the case whether or not the search directions are mutually conjugate. We mention that the paper of Sutti (1983) contains a mistake which will be discussed in section 2.

The method of Han (1986) is based on conjugate directions and is designed for parallel implementation. His approach is to construct conjugate subspaces whose direct sum is $\mathbb{R}^n$ and update these subspaces at each iteration. Suppose x^i is the current estimate and $T_1,\ldots,T_m$ are conjugate subspaces. Then for each i, $i = 1,\ldots,n$, we find a vector d_i which minimizes $f(x^i+d)$, $d \in T_i$, and take $d = d_1+\cdots+d_m$ as the search direction. The parallelism is therefore to find d_i's in parallel.

In the following sections, we shall investigate the case when we perform line searches from a current estimate along k linearly independent directions, in particular,

mutually conjugate directions, where k is less than or equal to the dimension of the problem.

2. Search in k directions.

In this section, we classify types of search when we are given k preassigned directions and investigate their properties in case of a convex quadratic function.

We consider an unconstrained minimization problem

$$\text{minimize } f(x),\ x \in \mathbb{R}^n, \tag{2.1}$$

where f is twice continuously differentiable and convex. Suppose we are given a set of k linearly independent search directions $d_1,...,d_k$, $k \le n$. Let x_c be the current estimate solution of (2.1). Then our subproblem is to find the new estimate denoted by x_+ in the linear variety $x_c + V_k$, where V_k is the subspace spanned by $d_1,...,d_k$, such that

$$f(x_+) < f(x_c). \tag{2.2}$$

We distinguish three ways of defining x_+.

(i) Global search: Find scalars $\alpha_1^G,...,\alpha_k^G$ such that the new estimate,

$$x_+^G = x_c + \alpha_1^G d_1 + \cdots + \alpha_k^G d_k, \tag{2.3}$$

minimizes $f(x)$, $x \in x_c + V_k$, i.e., $f(x_+^G) = \min_{x \in x_c + V_k} f(x)$.

(ii) Sequential search: Let $x_0^s = x_c$. For $i = 1,...,k$, find a scalar α_i^s which minimizes $f(x_{i-1}^s + \alpha d_i)$, $\alpha \in \mathbb{R}$. Then set the new estimate

$$x_+^s = x_k^s = x_c + \alpha_1^s d_1 + \cdots + \alpha_k^s d_k. \tag{2.4}$$

The third possible type of search which is the purpose of our investigation is the following.

(iii) Parallel search: For $i = 1,...,k$, find the line minimum of f along the line $x_c + \alpha d_i$, and obtain α_i^p which minimizes $f(x_c + \alpha d_i)$, $\alpha \in \mathbb{R}$. All the k line searches start from x_c. Then set the new estimate

$$x_+^p = x_c + \alpha_1^p d_1 + \cdots + \alpha_k^p d_k. \tag{2.5}$$

We first investigate the properties of (i), (ii) and (iii) for the case when $f(x)$ is a quadratic function of the form,

$$f(x) = \tfrac{1}{2} x^T Q x - b^T x + c, \tag{2.6}$$

where b is a vector in $\mathbb{R}^n$, c is a scalar, and Q is an $n \times n$ symmetric, positive definite matrix.

Theorem 2.1. Let $f(x)$, given by (2.6), be the function to be minimizead, and let $\nabla f(x) = Qx - b$ denote its graident. Suppose k linearly independent search directions $d_1, d_2, \ldots, d_k$, $k \leq n$, are given. Denote Qd_i by w_i, $i = 1,\ldots,k$. Then for $i = 1,\ldots,k$,

(a) $$(2.3) \Longleftrightarrow \alpha_1^G d_i^T w_1 + \cdots + \alpha_k^G d_i^T w_k = -d_i^T \nabla f(x_c), \tag{2.7}$$

(b) $$(2.4) \Longleftrightarrow \alpha_i^s = -\left[d_i^T \nabla f(x_c) + \sum_{j=1}^{i-1} \alpha_j^s d_i^T w_j \right] / (d_i^T w_i), \tag{2.8}$$

(c) $$(2.5) \Longleftrightarrow \alpha_i^p = -(d_i^T \nabla f(x_c)) / (d_i^T w_i). \tag{2.9}$$

The proof is straightforward and will be omitted.

Using the result of this theorem, we can easily obtain the following connection with some classical iterative methods.

Corollary 2.1. Suppose that $k = n$ and that $d_1, \ldots, d_n$ from theorem 2.1 are the standard unit vectors of $\mathbb{R}^n$. Then

(a) $$(2.3) \Longleftrightarrow Qs = -\nabla f(x_c), \text{ where } s = x_+ - x_c \tag{2.10}$$

(b) $$(2.4) \Longleftrightarrow \text{Gauss–Seidel for solving } Qx = b, \tag{2.11}$$

(c) $(2.5) \Longleftrightarrow$ Gauss–Jacobi for solving $Qx = b$ (2.12)

In general, x_+^G, x_+^s and x_+^p defined by (2.3)–(2.5) are different. Sutti (1983) takes a complete set of linearly independent search directions at each iteration, and shows that the points obtained by sequential and parallel searches are different. They are identical if the search directions are mutually conjugate. Here, we generalize for searching in k directions, $k \leq n$. Furthermore, we shall show that the converse is also true, that is, if $x_+^p = x_+^s$ for any x_c then those search directions are mutually conjugate.

Theorem 2.2. Under the hypothesis of theorem 2.1, the following statements are equivalent.

(a) $d_i^T Q d_j = 0$, $i,j \leq k$ and $i \neq j$.

(b) $x_+^s = x_+^G$ for any x_c.

(c) $x_+^p = x_+^G$ for any x_c.

(d) $x_+^s = x_+^p$ for any x_c.

Proof. If (a) holds, it follows from (2.7), (2.8) and (2.9) that $\alpha_i^G = \alpha_i^s = \alpha_i^p$, $i = 1,...,k$. Hence $x_+^G = x_+^s = x_+^p$ for any x_c. Now, we only need to show that (d) implies (a). Again, we denote Qd_i by w_i, $i = 1,...,k$. We prove the conjugate conditions,

$$d_m^T Q d_j = d_m^T w_j = 0, \quad 1 \leq j < m \leq k, \tag{2.13}$$

by induction. They are true when $m = 1$. Assume they hold for $m-1$. Since $x_+^s = x_+^p$, the linear independence of d_i's implies

$$\alpha_i^s = \alpha_i^p, \quad i = 1,...,k. \tag{2.14}$$

Using (2.8), (2.9) and (2.14), we have

$$\sum_{j=1}^{m-1} \alpha_j^s d_m^T w_j = \sum_{j=1}^{m-1} \alpha_j^p d_m^T w_j = \sum_{j=1}^{m-1} -\left[\frac{d_j^T \nabla f(x_c)}{d_j^T w_j} \right] (d_m^T w_j) = 0, \qquad (2.15)$$

which holds for any x_c. Since Q has full rank, we can choose x_c such that for a given w_ℓ, $1 \le \ell \le m-1$,

$$\nabla f(x_c) = w_\ell.$$

It follows from the last equation in (2.15) that

$$\sum_{j=1}^{m-1} \frac{(d_j^T w_\ell)(d_m^T w_j)}{(d_j^T w_j)} = 0. \qquad (2.16)$$

From the induction hypothesis and (2.16) we have that

$$d_m^T w_\ell = 0, \quad 1 \le \ell < m \le k.$$

The proof is complete. □

The above theorem assures that parallel search along mutually conjugate directions with respect to the Hessian of a convex quadratic function produces an x_+^p identical to x_+^G and therefore

$$f(x_+^p) = \min_{x \in x_c + V_k} f(x). \qquad (2.17)$$

For convex nonquadratic functions (2.17) does not hold in general. However, the point

x_+^p, can still be used to obtain a reduction in the objective function. Indeed we have

Theorem 2.3. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice continuously differentiable convex function and let $\nabla f(x)$, $\nabla^2 f(x)$ denote the gradient and the Hessian of f at x, respectively. Let x_c be the current approximation of the minimizer of $f(x)$ and suppose there are k linearly independent vectors $d_1,...,d_k \in \mathbb{R}^n$ such that

$$\nabla f(x_c)^T d_i < 0, \quad i = 1,...,k, \tag{2.18}$$

and

$$d_i^T \nabla^2 f(x_c) d_j = 0, \quad i \neq j. \tag{2.19}$$

Let x_+^p be defined as in (2.5) and let λ be a given number from the open interval $(0,1)$. Then the following algorithm

(*) IF $f(x_+^p) < f(x_c)$ THEN stop
ELSE $x_+^p \leftarrow x_c + \lambda(x_+^p - x_c)$ & GO TO (*)

terminates in at most $\text{Int}\left[1 - \frac{\ell n\ k}{\ell n\ \lambda}\right]$ steps.

Proof. From our hypothesis we have clearly

$$\alpha_i > 0, \ i = 1,...,k.$$

Therefore if we denote $d = x_+^p - x_c = \sum_{i=1}^{k} \alpha_i^p d_i$ then $\nabla f(x_c)^T d < 0$. This, together with the convexity of f shows that the function

$$\phi(t) = f(x_c + td) - f(x_c), \quad t > 0,$$

satisfies

$$0 < s \leq t \ \& \ \phi(t) < 0 \Longrightarrow \phi(s) < 0. \tag{2.20}$$

To see this, one has to note that ϕ is a convex function of a real variable for which

$$\phi(0) = 0\,, \quad \phi'(0) = \nabla f(x)^T d < 0.$$

From the convexity of f we deduce that

$$f\Big[\sum_{i=1}^{k} (1/k)(x_c + \alpha_i^p d_i)\Big] \le (1/k) \sum_{i=1}^{k} f(x_c + \alpha_i^p d_i)$$

$$\le \max_{1\le i\le k} f(x_c + \alpha_i^p d_i) < f(x_c), \tag{2.21}$$

which implies

$$\phi(1/k) < 0. \tag{2.22}$$

From (2.20) and (2.22) we have

$$\phi(s) < 0 \quad \text{whenever} \quad 0 < s \le (1/k). \tag{2.23}$$

Let us denote $x_{+,0}^p = x_+^p$ given by (2.5) and

$$x_{+,m+1}^p = x_c + \lambda(x_{+,m}^p - x_c).$$

It follows that

$$x_{+,j}^p = x_c + \lambda^j(x_+^p - x_c) = x_c + \lambda^j d.$$

Finally, by using (2.23) we obtain

$$f(x_{+,j}^p) < f(x_c) \quad \text{whenever} \quad j \ge -\frac{\ell n\ k}{\ell n\ \lambda}.$$

The proof is complete. □

We note that for $k = n$ (2.21) reduces to a relation used by Sutti (1983). Sutti also claims that for any convex function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$,

$$f\Big[x_c + (1/n)\sum_{i=1}^{n} \alpha_i^p d_i\Big] \le (1/n) f\Big[x_c + \sum_{i=1}^{n} \alpha_i^p d_i\Big]. \tag{2.24}$$

This is clearly false for $n \ge 2$ since in case of a positive and convex quadratic function $f(x)$

and $d_1,\ldots,d_n$ are conjugate with respect to $\nabla^2 f(x_c)$, (2.24) would imply

$$0 < f\Big[x_c+(1/n)\sum_{i=1}^{n}\alpha_i^p d_i\Big] \le (1/n) f\Big[x_c+\sum_{i=1}^{n}\alpha_i^p d_i\Big]$$

$$= (1/n)\min_{x\in\mathbb{R}^n} f(x) \le (1/n) f\Big[x_c+(1/n)\sum_{i=1}^{n}\alpha_i^p d_i\Big],$$

which in turn implies that $1 \le (1/n)$.

3. A parallel line search algorithm.

Our algorithm is composed of the following five basic steps:

Set $i = 0$, x_0 = an approximation of the minimizer of $f(x)$.

A. Find mutually conjugate directions $d_1,\ldots,d_k$ with respect to $\nabla^2 f(x_i)$.

B. Perform parallel line search along $d_1,\ldots,d_k$, all starting from $x_c = x_i$ to obtain $\alpha_1^p,\ldots,\alpha_k^p$.

C. Set $x_{i+1} = x_i + \sum_{j=1}^{k}\alpha_j^p d_j$.

D. IF $f(x_{i+1}) < f(x_i)$ GO TO E

ELSE $x_{i+1} \longleftarrow x_i + \lambda(x_{i+1} - x_i)$ & GO TO D. (3.1)

E. IF a certain stopping criterion is satisfied

THEN return x_{i+1} as the desired approximation of the minimizer and STOP

ELSE $i \longleftarrow i+1$ and GO TO A.

Now let us give some details on step A. The generation of conjugate directions is done by using some quasi–Newton updates (see Dennis and Schnabel (1983) for the properties of such updates). We have

Given $B_c \in \mathbb{R}^{n\times n}$. Set $B_1 = B_c$.

Do for $i = 1,\ldots,k$.

A.1 $$d_i = -B_i\nabla f(x_c) + \sum_{j=1}^{i-1} \bar{d}_j^T \nabla f(x_c)\bar{d}_j. \tag{3.2}$$

A.2 $$w_i = \nabla^2 f(x_c)d_i\,, \quad \sigma_i = \sqrt{d_i^T w_i}$$
$$\bar{d}_i = d_i/\sigma_i\,, \quad \bar{w}_i = w_i/\sigma_i. \tag{3.3}$$

A.3 Update B_i to obtain B_{i+1} so that

$$\bar{w}_j^T B_{i+1} = \bar{d}_j^T\,, \quad j = 1,\ldots,i. \tag{3.4}$$

Update formulas can be any of the following:

A.3.1. $$B_{i+1} = B_i + \bar{d}_i(\bar{d}_i^T - \bar{w}_i^T B_i),$$

A.3.2. $$B_{i+1} = B_i + \frac{B_i\bar{w}_i(\bar{d}_i^T - \bar{w}_i^T B_i)}{\bar{w}_i^T B_i \bar{w}_i},$$

A.3.3. $$B_{i+1} = B_i + \frac{(\bar{d}_i - B_i\bar{w}_i)(\bar{d}_i - B_i\bar{w}_i)^T}{(\bar{d}_i - B_i\bar{w}_i)^T\bar{w}_i},$$

A.3.4. $$B_{i+1} = B_i + \bar{d}_i\bar{d}_i^T - \frac{B_i\bar{w}_i\bar{w}_i^T B_i}{\bar{w}_i^T B_i \bar{w}_i},$$

$$\text{A.3.5.} \quad B_{i+1} = B_i + (\bar{d}_i - B_i\bar{w}_i)\bar{d}_i^T + \bar{d}_i(\bar{d}_i - B_i\bar{w}_i)^T - \bar{w}_i^T(\bar{d}_i - B_i\bar{w}_i)\bar{d}_i\bar{d}_i^T ,$$

or for that matter any update formula which satisfies (3.4). For updates (A.3.2) and (A.3.3) we can replace $\bar{d}_i$ and $\bar{w}_i$, by d_i and w_i respectively. Indeed it is easy to verify that if (3.4) is satisfied then the algorithm A.1–A.3 generates directions $d_1,\ldots,d_k$ which are conjugate with respect to $\nabla^2 f(x_c)$.

In this scheme, the vector $w_i = \nabla^2 f(x_c)d_i$ needs to be computed. Avoiding the computation of the Hessian, we use finite differences by gradients (if available) to approximate

$$w_i = (1/h)(\nabla f(x_c + hd_i) - \nabla f(x_c)), \tag{3.5}$$

provided $d_i^T w_i > 0$. We also make use of the information about the changes in the gradients, that is, x_c, x_+, $\nabla f(x_c)$ and $\nabla f(x_+)$ are used to update B_k at the end of each iteration, employing the same update formula, to obtain B_c for the next iteration. At iteration 0, we initialize B_c with I or use the initialization similar to Shanno–Phua for the direct update suggested in Dennis and Schnabel (1983). For the latter initialization, we need to perform one line search at iteration 0 along $-\nabla f(x_0)$, if x_0 is the starting point, to obtain x_+, then we compute $y = \nabla f(x_+) - \nabla f(x_c)$ and set

$$B_c = \gamma I , \tag{3.6}$$

where $\gamma = \dfrac{y^T(x_+ - x_0)}{y^T y}$.

Schnabel (1987) suggests the efficient use of the available processors while doing line search, that is, when one processor is computing the function value at the trial point, the available processors can compute concurrently some of the components of the gradient at this trial point. If this trial point is acceptable then only the remaining components need to be computed. Lootsma and Ragsdell (1988) suggest parallel evaluations of the function at the grid–points in the located interval in the line search. Therefore, besides doing line

searches in parallel, we can further use the available processors to perform simultaneous function or gradient evaluations within each line search itself, especially when k, the number of search directions, is much smaller than the number of processors.

We can now give our algorithm which is implemented and run with some problems presented in section 4.

Algorithm 3

Given $f : \mathbb{R}^n \to \mathbb{R}$, x_0 = starting point, $B_0 \in \mathbb{R}^{n \times n}$ (symmetric, positive definite), k = number of search directions.

step 1. Initialization

$$x_c = x_0,\ f_c = f(x_0),\ g_c = \nabla f(x_0),\ B_c = B_0$$

step 2. Stop if any stopping criterion is satisfied

(* iteration *)

step 3. Generate mutually conjugate directions $d_1,\ldots,d_k$

Set $B_1 = B_c$

Do for $i = 1,\ldots,k$

$$d_i = -B_i g_c + \sum_{j=1}^{i-1} \bar{d}_j^T g_c \bar{d}_j$$

Use finite differences to obtain

$$w_i = (1/h)(\nabla f(x_c + h d_i) - g_c)$$

Normalize

$$\sigma_i = \sqrt{d_i^T w_i},\ \bar{d}_i = d_i/\sigma_i,\ \bar{w}_i = w_i/\sigma_i$$

Update B_i (skip the update if $i = k$).

step 4. Perform parallel line searches along $d_1,...,d_k$, all starting from x_c to obtain $\alpha_1^p,...,\alpha_k^p$ (exact or inexact line search).

step 5. Set $x_+ = x_c + \sum_{i=1}^{k} \alpha_i^p d_i$.

step 6. If $f(x_+) < f(x_c)$ then go to step 7
else $x_+ \leftarrow x_c + \lambda(x_+ - x_c)$, $0 < \lambda < 1$, and go to step 6.

step 7. Test for convergence: if the stopping criterion is satisfied then stop
else go to step 8.

step 8. Update B_k from step 3 by using x_c, x_+, g_c, $\nabla f(x_+)$. Set $d = x_+ - x_0$, $w = \nabla f(x_+) - g_c$ and use the same update formula as in step 3. Obtain B_c.

step 9. Set $x_c = x_+$, and $g_c = \nabla f(x_+)$.
Go to step 2.

4. Some numerical results.

For the purpose of numerical experimentation, we first run algorithm 3 with a simple convex function of variable dimension,

$$f(x) = \frac{1}{2} x^T A x - b^T x + \frac{1}{4} \sum_{i=1}^{n} (x)_i^4, \tag{4.1}$$

where $A = (a_{ij})$ is a symmetric diagonally dominant $(n \times n)$ matrix having $a_{ii} = 1$, $a_{ij} = (0.5)^{j-i+1}$ for $j > i$, b is a constant vector computed so that the minimum of the quadratic $\frac{1}{2} x^T A x - b^T x$ is $(1,1,1,...)$, and $(x)_i$ is the i^{th} component of x.

In our implementation, the initialization of the update matrix B is given by (3.6), and λ in step 6, algorithm 3 is set to be 0.9. Two line search methods are implemented, one is the backtracking method using quadratic fit and cubic fit, and the other is the exact

line search, secant method, solving $\nabla f(x + \alpha d)^T d = 0$. See Dennis and Schnabel (1983) for details on the inexact line search procedure. Update formulas A.3.3–A.3.5, in section 3, are implemented to compare their performances. For comparison, we also run the unconstrained minimization code, UNCMIN, developed by R.B. Schnabel and collaborators (University of Colorado, Boulder), which is based on quasi–Newton methods if the analytic Hessian is not available or the function is expensive to evaluate, otherwise on Newton's method. Both our algorithm and UNCMIN are run with double precision and with the same stopping criterion. For the problem (4.1), the starting point is $x_0 = (0,1,0,1,...)$, and we run our algorithm and UNCMIN for three tries. The starting point in the first try is x_0. For the second and the third try the starting points are $10x_0$ and $100x_0$, respectively.

Table 4.1 presents the results obtained by UNCMIN and algorithm 3 which uses update formula A.3.3 (symmetric, rank 1 update), and the number of search directions, $k = 2$. The exact line search gives good performance here, but if the function or gradient evaluation is expensive, this may not be practical. The effect of searching in 2 directions is not very evident when the starting point is x_0 or in the first try, but it is revealing for the second and third tries, compared with the quasi–Newton method. For this problem (4.1), the increase in the dimension does not affect the number of iterations needed for convergence.

We further run algorithm 3 with the variable dimension test problem of Moré, Garbow and Hillstrom (1981). The two test problems are Chebyquad function of dimension < 50, and Trigonometric function with variable dimension. We use the update formula A.3.3 and inexact line search in our algorithm for both problems. Results are presented in Table 4.2. Both UNCMIN and algorithm 3 give the same solution. The reduction in the number of iterations is almost half when $k = 2$, for example, Chebyquad function, dimension = 32, it requires 93 iterations when $k = 1$ and 55 iterations when

$k = 2$, but when $k = 3$ it requires 51 iterations. We have used the standard starting points and have run for only one try.

Table 4.1

dimension	UNCMIN (quasi–Newton) iter.	Alg. 3, search in 2 directions inexact line search iter.	exact line research iter.
n=8	7	5	4
n=8	27	12	7
n=8	61	27	9
n=16	7	5	3
n=16	27	14	6
n=16	63	30	9
n=32	6	5	3
n=32	25	12	8
n=32	61	29	9
n=64	6	4	3
n=64	25	12	6
n=64	58	29	9

Table 4.2

dimension	UNCMIN (quasi–Newton) iter.	Algorithm 3 k=1 iter.	k=2 iter.	k=3 iter.	
n=8	26	25	12	11	
n=16	32	29	18	21	Chebyquad
n=32	112	93	55	51	
n=8	22	19	13	11	
n=16	34	29	25	18	Trigonometric
n=32	34	33	20	15	

From these numerical results, we see that the number of search directions should not be too large. Therefore, while searching in parallel, the available processors can be used to perform simultaneous function, gradient evaluations if needed in the line search method used. We hope that further numerical experiments and theoretical investigations will lead to an improvement of algorithm 3 so that it will run efficiently on shared memory parallel machines composed of a relatively small number of processors.

REFERENCES

D. Chazan and W.L. Miranker (1970), "A nongradient and parallel algorithm for unconstrained minimization," SIAM J. Control 8, 207–217.

J.E. Dennis and R.B. Schnabel (1983), "Numerical methods for unconstrained optimization and nonlinear equations," Prentice–Hall, Englewood Cliffs, N.J.

S.P. Han (1986), "Optimization by updated conjugate subspaces," in Numerical Analysis, Pitman Research Notes in Mathematics Series 140, D.F. Griffith and G.A. Watson, eds., Longman Scientific and Technical, Burnt Mill, England, 82–97.

T.A. Lootsma and K.M. Ragsdell (1988), "State of the art in parallel nonlinear optimization," Parallel Computing 6, North–Holland, 133–155.

J.J. Moré, B.S. Garbow, and K.E. Hillstrom (1981), "Testing unconstrained optimization software," ACM Transactions on Mathematical Software 7, 17–41.

M.J.D. Powell (1964), "An efficient method for finding the minimum of a function of several variables without calculating derivatives," Computer J. 7, 155–162.

R.B. Schnabel (1987), "Concurrent function evaluations in local and global optimization," Computer Methods in Applied Mechanics and Engineering 64, 537–552.

C. Sutti (1983), "Nongradient minimization methods for parallel processing computers," JOTA 39, 465–488.