A new Parallel VM algorithm for solving large scale optimizations problems.

Abbas Y. Al-Bayati **Muna M. Mohammed Ali**
Department of Mathematics
College of Computers sciences and Mathematics
University of Mosul

Omar B. Mohammed
Department of mathematics
KOYA University

Received **Accepted**
16 / 02 / 2009 **27 / 07 / 2009**

الخلاصة

في هذا البحث تم استحداث خوارزمية جديدة في حقل الخوارزميات المتوازية للمتري المتغير بالاعتماد على شروط (Wolf-powell) وباستخدام (32) دالة لا خطية مع الحصول على نتائج مشجعة مقارنة بالخوارزمية القياسية.

Abstract

In this paper, a new optimal parallel line search step-size is designed to improve the parallel VM algorithm and satisfies Wolf's-Powell condition by using thirty-two non-linear test problems. The new proposed algorithm has been worked well on our selected test problems, and it has a superiority on the standard algorithm.

1. Introduction

The parallelization of QN methods has been considered by several researchers in the past decades. The first work was done by Satraeter (1973) for the symmetric rank-one method and it was later modified by Van Laarhoven (1985). At each step, this algorithm updates the Quasi-Newton matrix along m independent directions by parallelly evaluating values of the functions and the gradients at m points.

For a positive definite quadratic function, it can be shown that the method converges in one iteration regardless of the initial starting point. Computational results of Laarhoven (1985) for a set of well-known

problems with no more than four variables indicates that the algorithm improves the total number of parallel function evaluations, and the total number of iterations.

In the well known variables metric algorithms, we start from an arbitrary point $x^{\circ} \in E^n$. The search direction s^k at the k-th iteration of the algorithm is constructed through the following relation:

$$s^k = -H_k \Delta f(x^k) \ldots\ldots\ldots \quad (1)$$

where f is the function to be minimized (assumed to have continuous second derivatives),x^k the current iteration point and H_k an approximation to the inverse Hessian matrix at x^k.

The next iteration point x^{K+1} is obtained by

$$x^{K+1} = x^K + \lambda_K s^K \ldots\ldots\ldots \quad (2)$$

where λ_k stands for an approximation to a minimum of along the search direction s^k from x^k. Finally, we update H_k by an additive correction D_k, yielding

$$H_{k+1} = H_k + D_k \ldots\ldots\ldots \quad (3)$$

where D_k depends on

$$\sigma^k = x^{k+1} - x^k \ldots\ldots\ldots \quad (4)$$

and

$$Y^k = \Delta f(x^{k+1}) - \Delta f(x^k) \ldots\ldots\ldots \quad (5)$$

There are many possible choices for the correction matrix D_k. We consider first the so-called rank-two updating formula (Fletcher, 1980) where D_k is given by

$$D_k = \frac{H_k y^k (y^k)^T H_k}{(y^k)^T H_k y^k} + \frac{\sigma^k (\sigma^k)^T}{(\sigma^k)^T y^k} + W_k W_k^T \qquad \ldots\ldots\ldots \quad (6)$$

where

$$W_k = ((y^k)^T H_k y^k)^{1/2} \left[\frac{\sigma^k}{(\sigma^k)^T y^k} - \frac{H_k (y^k)^T}{(y^k)^T H_k y^k} \right]$$

and we update H^k with conjugate gradient method (CG) method by calculating the search direction S^{k+1} as:

$$s^{k+1} = -H_{k+1} \Delta f(x^{k+1}) + \frac{(y^k)^T H_{k+1} \Delta f(x^{k+1})}{(y^k)^T s^k} . s^k \ldots\ldots\ldots \quad (7)$$

where H_{k+1} is any positive-definite symmetric matrix, or preconditional CG method (PCG).

The search direction (defined in (7) was first proposed by Nazareth (1979) and was also called the preconditioned Hestense and Stiesfel method. It has been shown that for a quadratic q as given by:

$$q(x) = \frac{1}{2} x^T A x - b^T x \ldots\ldots\ldots \quad (8)$$

It must be true that

$H_1 = A^{-1}$

Thus, it takes only one linear search to find the minimum of q, for arbitrary initial point $x^{\circ} \in E^n$, since the first search direction is the Newton direction. A sequential variable metric method, however, needs n linear searches. We expect a similar reduction in linear searches for the non-quadratic case. This is extremely important, since linear searches take a large amount of function evaluations. A further reduction in computing time can be achieved by parallelizing the linear search itself. (see Van Laarhoven, 1985).

2. Standard Parallel Al-Bayati and Aaref2004Self-Scaling Algorithm:

Step 1: for any sarting point x_o, and initial matrix H_0 (usually H_0=I) and n linearly $\delta^1, \delta^2, \ldots, \delta^n$,set k=0.

Step 2: let $s^1 = -H_1 \Delta f(x1)$

Step 3: for k=1,…. n, $x^{K+1} = x^K + \lambda_K s^K$ …where λ_K argmin f($x^K + \lambda_K s^K$)=ELS (exact line search; sequential wolf line search conditions is used).

Step 4: check if $\| \Delta f(x^{k+1}) \| < \in$, is small number, stop otherwise, continue.

Step 5: calculate in parallel $\Delta f(x^k)$, $\Delta f(x^{k1})$, …, $\Delta f(\Delta^{kn})$, where $x^{kj} = x^k + \delta^j$ and $y^{kj} = \Delta f(x^{kj}) - \Delta f(x^k)$

Compute γ^{kj}, j = 1, …, n through

$$\gamma^{kj} = \frac{\delta^j}{(y^{kj})^T s^j} - \frac{H_k y^{kj}}{(y^{kj})^T H_k y^{kj}}$$ and update H_k through

$$H_{k+1} = H_k - \frac{H_k y^{kj} (y^{kj})^T H_k}{(y^{kj})^T H_k y^{kj}} + \frac{\delta^j (\delta^j)^T}{(\delta^j)^T y^{kj}} + \left[(y^{kj})^T H_k y^{kj} \right]^{1/2} \gamma^{kj} (\gamma^{kj})^T \qquad (9)$$

Step 6: develop search direction

$$S^{k+1} = H_k g_{k+1} + \frac{g_{k+1} H_k g_{k+1}^T}{g_{k+1} g_{k+1}^T}$$

Where $g_{k+1} = \Delta f(x^{k+1})$

Step 7: if available storage is exceeded then employ a restart option either with k = n or with powell switching criterion or $g_{k+1}^T H_k g_{k+1} > 0$ where $g_{k+1}^T H_k g_{k+1} > 0$ guarantee the positive definiteness of H.

For more details see Al-Bayati &Aaref (2004).

3. New Parallel VM algorithm

The proposed parallel VM(PVM) algorithm consists of the following steps:

1) Initialization: let k= 0, and $x_\circ$ be the initial guess of the minimum and $H_\circ$ = I be the identify matrix. Set ε > 0 as the required accuracy.
2) compute the function and gradient values at x_k
   let $f_k = f(x_k)$
   and $g_k = \nabla f(x_k)$

3) compute the parallel search directions. Let $m_1 > 0$ be the number of processors available for computing the search directions in parallel compute

$$d_k^j = -H_k(t_j)g_k,\ j=1,2,...,m_1 \quad \ldots\ldots\ldots \quad (10)$$

In this paper, we apply the parallel line search algorithm for the Al-Bayatiy (1991) VM-algorithm: Proceed as follows:

Call the line search routine in parallel along each search direction $d_k^j,\ j=1,2,...,m_1$, stop executing this procedure once a line minimum λ_k has been found to satisfy the following Wolf's condition along any search direction d_k^j:

$$f(x_k + \lambda_k d_k^j) \le 0.0001 \text{ x } \lambda_k\ g_k^T\ d_k^j \quad \ldots\ldots\ldots \quad (11)$$

And

$$\nabla f(x_k + \lambda_k d_k^j) \ge 0.09 \text{ x } g_k^T\ d_k^j \ldots\ldots\ldots \quad (12)$$

Let d_k^* be the search direction that α_k has been found successfully.

In this step, if a line minimum points are found from more than one search direction, then d_k^* is choses to be the search direction that attains the lowest minimum point. The details of our new line searches used in our proposed parallel algorithm is given bellow:

3.1 Parallel Line Search Step (PLS)

The parallel line-search procedure works as follows. Let m_2 be the number of parallel processors available for locating the minimum along a particular search direction $d_k^j,\ j=1,2,...,m_1$, let λ_{max} be the maximum allowable step-size, and denote $\psi(\lambda) = f(x_k + \lambda\ d_k^j)$. The parallel line search process consists of the following steps:

1. Choose the step sizes:

Let $0 < \lambda^j < \lambda_{max}$, $i = 1, 2, \ldots m_2$ where $\lambda^1 < \lambda^2 < \lambda^{(m_2)}$ are m_2 different approximately chosen step sizes. For instance, we may chose $\lambda^1 = 0.5$, $\lambda^2 = 1.0$, … , $\lambda^{m2} = \lambda_{max}$, let Φ be the set of these step sizes.

3.2 Compute the function and gradient values concurrently

For $i = 1, 2, \ldots, m_2$, compute concurrently

$$x_k^i = x_k + \lambda^i d_k^j,$$

$$f_k^i = f(x_k^i),$$

$$g_k^i = \nabla f(x_k^i)$$

3. Test for successful points

Let Φ^* be the set of step sizes λ^i such that for each $\lambda^i \in \Phi^*$, λ^i satisfies the Wolf's conditions (2.3) – (2.4). If $\Phi^* \neq \phi$ (empty set) and $\lambda_k^i \in \Phi^*$ is the step size which corresponds to the minimum functional value, that is,

$$\Psi(\lambda_k^i) = \min_{\lambda^i \in \Phi^+} f(x_k + \lambda^i d_k^j)$$

Then set $\lambda_k = \lambda_k^i$ and return to the main PQN routine; otherwise, proceed to step 4.

3.3 Choose interpolation points:

Let Φ^+ be the set of step sizes such that for each $\lambda^i \in \Phi^+$, λ^i satisfies $d_k^{jT} g_k^i > 0$.

Let $\Phi^- = \Phi - \Phi^+$. Choose $\lambda_1 \in \Phi^-$ such that

$$\Psi(\lambda_1) = \min_{\lambda^i \in \Phi^-} f(x_k + \lambda^i d_k^j)$$

And choose $\lambda_2 \in \Phi^+$ such that

$$\Psi(\lambda_2) = \min_{\lambda^i \in \Phi^+} f(x_k + \lambda^i d_k^j)$$

If $\Phi^- = \Phi$, then choose $\lambda_1 = 0$. If $\Phi^+ = \Phi$, then choose $\lambda_2 = \lambda^m$, where $\lambda^m \in \Phi^-$ such that

$$\Psi(\lambda^m) = \min_{\lambda^i \in \Phi^-} f(x_k + \lambda^i d_k^j)$$

3.4 Apply the cubic interpolation technique

Let $\varphi(\lambda)$ be the cubic polynomial passing the two points λ_1 and λ_2. Let λ^* be the minimum of $\varphi(\lambda)$

Compute $\varphi(\lambda^*) = f(x_k + \lambda^* d_k^j)$

and

$h^* = \nabla\psi(\lambda^*)$

If λ^* satisfies Wolfe's conditions(2.3)–(2.4),then set $\lambda_k = \lambda^*$ and return to the main PQN routine. Otherwise, if $d_k^{jT} . \nabla\psi(\lambda^*) > 0$ then replaces λ_1 with λ^*; if $d_k^{jT} . \nabla\psi(\lambda^*) \le 0$ then replace λ_2 with λ^*. Repeat step 5.

4. Outlines of the new proposed algorithm.

In this section, we are going to a new optimal step by modifying the parallel Al-Bayati self- scaling (1991) Algorithm by using parallel line search procedure (λ^i) and satisfies the Wolf's condition.

4.1 Outlines of the new parallel VM- algorithm.

Step 1: for any starting point x_o, and initial matrix (usually $H_o = I$), and n linearly $\delta^1, \delta^2, \ldots, \delta^k$, set k=0.

Step 2: let $s^1 = -H_1 \Delta f(x^1)$

Step 3: for k=1, …, n, i= 1, …, m, $x^{k+1} = x^k + \lambda_i s^k$

Where λ_i is parallel line search satisfies Wolf's condition $f(x_k + \lambda_k d_k^j) \le 0.0001 x \lambda_k g_k^T d_k^j$ and satisfies conditions (3.1,3.2,3.3)

Step 4: check if $\|\Delta f(x^{k+1})\| < \in$, $\in$ is small number, stop otherwise, continue.

Step 5: calculate in parallel $\Delta f(x^k)$, $\Delta f(x^{k1})$, …, $\Delta f(\Delta^{kn})$, where $x^{kj} = x^k + s^j$ and $y^{kj} = \Delta f(x^{kj}) - \Delta f(x^k)$, $Sc = \frac{(y^{kj})^T H_k y^{kj}}{(y^{kj})^T \delta^j}$

Compute γ^{kj}, j = 1, …, n through

$\gamma^{kj} = \frac{\delta^j}{(y^{kj})^T s^j} - \frac{H_k y^{kj}}{(y^{kj})^T H_k y^{kj}}$ and update H_k through

$$H_{k+1} = H_k - \frac{H_k y^{kj}(y^{kj})^T H_k}{(y^{kj})^T H_k y^{kj}} + sc\frac{s^j(s^j)^T}{(s^j)^T y^{kj}} + \left[(y^{kj})^T H_k y^{kj}\right]^{1/2}\gamma^{kj}(\gamma^{kj})^T$$

Where $Sc = \frac{(y^{kj})^T H_k y^{kj}}{(y^{kj})^T s^j}$

Step 6: develop search direction

$$S^{k+1} = H_k g_{k+1} + \frac{g_{k+1} H_k g_{k+1}^T}{g_{k+1} g_{k+1}^T}$$

Where $g_{k+1} = \Delta f(x^{k+1})$

Step 7: if available storage is exceeded then employ a restart option either with k = n or with powell switching criterion or $g_{k+1}^T H_k g_{k+1} > 0$ where $g_{k+1}^T H_k g_{k+1} > 0$ guarantee the positive definiteness of H.

5. Numerical Results and Conclusions:

The comparison tests involve well knowon test functions with different dimensions (Bunday, 1984). All the results were obtained using programs written in FORTRAN.

The comparative performance of the algorithms are evaluated by considering both total number of iteration (NOI) and total number of function (NOF). The stopping criterion is taken to be

$\|\Delta f(x^{k+1})\| < 5*10^{-5}$

The line search employed is the cubic fitting technique which uses function values and their gradients. Which is fully described in Bunday (1984).

Two algorithms were tested, namely

1) Standard Parallel Al-Bayati and Aaref(2004) Algorithm.
2) The new proposed algorithm.

Our numerical results are presented in two tables. Table(5.1) compares between the two algorithms using cubic fitting line search with Wolfe condition, for sixteen small dimensionally test functions $4 \leq n \leq 80$. It is clear that the idea of parallel algorithms is well defined in the field of parallel methods.

Namely, there are about (17)% NOI and (12)% NOF improvements on the standard Al-Bayati's*Aaref (2004) method. Also, table (5.2) represents the results of our numerical comparison between the two algorithms, but for sixteen large dimensionality test functions $100 \leq n \leq 500$. In fact the new algorithm in this case and for the selected set of

test functions has an improvement of about (19)% NOI and (15) NOF on the standard well known algorithm.

Table(5.1): Comparative of New-algorithm against the standard Al-Bayati&Aaref(2004)algorithm for 4≤n≤80

Test Function	n	Standard parallel Al-Bayati &Aaref(2004)		(new)	
		NOI	NOF	NOI	NOF
Rosen brock	4	8	27	4	21
Powell	4	19	43	19	43
Wood	4	14	33	14	33
Wolf	4	15	40	11	31
Helical	8	17	38	15	30
Cubic	10	17	44	9	41
Recipe	10	18	45	18	45
Miele	10	13	34	15	34
Miele	20	13	34	13	34
Edger	20	4	13	4	9
Tridigid	30	12	30	12	30
Helical	40	16	35	9	30
Wolf	40	40	80	30	72
Powel	40	15	34	11	25
Wood	60	15	34	11	25
Rosen	80	8	17	8	9
Total		244	581	203	512

Table(5.2): Comparative of New-algorithm against the standard Al-Bayati&Aaref (2004) algorithm for 100≤n≤500

Test Function	n	Standard parallel Bayati&Aaref(2004)		New	
		NOI	NOF	NOI	NOF
Wolfe	100	45	89	34	75
Cubic	100	11	37	11	37
Dixon	100	21	67	11	50
Powell	120	21	48	14	31
Rosen	120	12	35	10	32
Wood	120	18	35	17	35
Helical	160	10	20	10	20
Edger	160	10	21	6	11
Reciep	200	10	19	9	13
Miele	200	28	72	24	67
Tridigid	200	30	40	28	40
Rosen	240	12	35	10	30
Miele	300	30	72	24	67
Cubic	350	13	36	11	37
Reciep	400	4	13	9	13
Powell	500	22	48	14	31
Total		297	687	242	589

6. Appendix:

1- Generalized Powell Function:

$$f = \sum_{i=1}^{n/4}\left[\left(x_{4i-3} - 10x_{4i-2}\right)^2 + 5\left(x_{4i-1} - x_{4i}\right)^2 + \left(x_{4i-2} - 2x_{4i-1}\right)^4 + 10\left(x_{4i-3} - x_{4i}\right)^4\right],$$

$$x_0 = \left(3,-1,0,1;\cdots\right)^T.$$

2- Generalized Wood Function:

$$f = \sum_{i=1}^{n/4} 100\left[\left(x_{4i-2} - x_{4i-3}^2\right)^2\right] + \left(1 - x_{4i-3}\right)^2 + 9\left(x_{4i} - x_{4i-1}^2\right)^2 + \left(1 - x_{4i-1}^2\right)^2$$

$$+ 10.1\left[\left(x_{4i-2} - 1\right)^2 + \left(x_{4i} - 1\right)^2\right] + 19.8\left(x_{4i-2} - 1\right)^2\left(x_{4i} - 1\right),$$

$$x_0 = \left(-3,-1,-3,-1;\cdots\right)^T.$$

3- Generalized Sum of Quadratics Function:

$$f = \sum_{i=1}^{n}\left(x_i - 1\right)^4,$$

$$x_0 = \left(2;\cdots\right)^T.$$

4- Generalized Dixon Function:

$$f = \sum_{i=1}^{n}\left[\left(1 - x_i\right)^2 + \left(1 - x_n\right)^2 + \sum_{i=1}^{n-1}\left(x_i^2 - x_{i-1}\right)^2\right],$$

$$x_0 = \left(-1;\cdots\right)^T.$$

5- Generalized Rosenbrock Function:

$$f = \sum_{i=1}^{n/2}\left[100\left(x_{2i} - x_{2i-1}^2\right)^2 + \left(1 - x_{2i-1}\right)^2\right],$$

$$x_0 = \left(-1,2,1;\cdots\right)^T.$$

6- Generalized Cubic Function:

$$f = \sum_{i=1}^{n/2}\left[100\left(x_{2i} - x_{2i-1}^3\right)^2 + \left(1 - x_{2i-1}\right)^2\right],$$

$$x_0 = \left(-1,2,1;\cdots\right)^T.$$

7- Generalized Tri Function:

$$f = \sum_{i=1}^{n}\left(ix_i^2\right)^2,$$

$$x_0 = \left(-1;\cdots\right)^T.$$

References

1) Al-Bayati, A. Y., 1991. "A new family of self-scaling variable metric algorithms for unconstrained optimization", Journal of Education on Science, Mosul University, 12, p. 25-32.

2) Al-Bayati, A.Y. & Aaref (2004). "New Preconditioned parallel BFGS algorithm for optimization", Raf. Jour. Sc.Vol.15, No.1, P.52-59.

3) Bunday, B., 1984. "Basic optimization methods", Edward Arnold, London.

4) Byrd R.H., Schnabel R.B. and Shultz G.A., (1988), "Parallel quasi-Newton methods for unconstrained optimization", Mathematical Programming, pp. 273-306.

5) Fletcher, R., 1980., "Practical methods of optimization", Volume 1, Unconstrained optimization, John Wiley and sons, New York and Toronto.

6) Luksan L., (1994), "Computational experience with known variable metric updates", Journal of Optimization Theory and Applications, pp. 27-47.

7) Nag (1992): "Fortan Library Reference manual (Mark 14), Numerical Algorithms Group 3", Nag Limited, Oxford OX2-8DR, (1992), United Kingdom.

8) Nazareth, L., 1979. "A relationship between the BFGS and conjugate algorithms and implications for new algorithms", SIAMJ. Number, Anal., 16. pp. 794-800.

9) Paul Kang Hoh Phua, Weigue Fan, Daohua Ming. Department of Information Systems & Computer Science. National University of Singapore (2007). Parallel Alg. for large scale nonlinear optimization.

10) Straeter, T.A. 1973. "A parallel variable metric optimizatiom algorithms", NASA. Technical note D-7329 Hamptron, Virginia, pp.236-650.

11) Van Laarhoven, P., 1985., "Parallel variable metric algorithms for unconstrained optimization", Mathematical programming 33, 68-81.