

Self-Scaling Parallel Quasi-Newton Methods

Paul Kang-Hoh PHUA, Weiguo FAN, Yuelin ZENG

Department of Information Systems and Computer Science
National University of Singapore
Lower Kent Ridge Road, Singapore 119260
Email: (*phuakh, fanweigu, zengyuel*)@iscs.nus.sg

Abstract

In this paper, a new class of self-scaling quasi-Newton(SSQN) updates for solving unconstrained nonlinear optimization problems(UNOPs) is proposed. It is shown that many existing QN updates can be considered as special cases of the new family. Parallel SSQN algorithms based on this class of class of updates are studied. In comparison to standard serial QN methods, proposed parallel SSQN(SSPQN) algorithms show significant improvement in the total number of iterations and function/gradient evaluations required in solving a wide range of test problems. In fact, the average speedup factors by the new SSPQN algorithms over the conventional BFGS method and E04DGE in NAG library are 3.22/3.13 and 2.80/3.09 , respectively(in terms of total number of iterations and total number of function/gradient evaluations required). For some test problems, the speedup factor gained by the new algorithms can be as high as 25/25 over BFGS and 20/25 over E04DGE , both in terms of total number of iterations and function/gradient evaluations.

1 Introduction

We are concerned here with the numerical methods for solving the following unconstrained nonlinear minimization problem(UNOP)

$$\min_{x \in R^n} f(x) \tag{1.1}$$

where the objective function $f(x)$ is a twice continuously differentiable, real valued function defined in n-dimensional space .

A popular class of methods for solving problem (1.1) is the quasi-Newton(QN) methods. Given an initial point x_1 , an $n \times n$ matrix H_1 , QN methods proceed to generate the following iterative sequence at the k_{th} iteration:

1. Compute $g_k = \nabla f(x_k)$, the gradient of the function $f(x)$ at the current point x_k ;
2. Compute the search direction

$$d_k = -H_k g_k \quad (1.2)$$

3. Apply an appropriate line search strategy along the search direction d_k to find a step-size $\alpha_k > 0$, such that the following Wolfe's condition are satisfied:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \beta_1 \alpha_k g_k^T d_k \quad (1.3)$$

$$g(x_k + \alpha_k d_k)^T d_k \geq \beta_2 g_k^T d_k \quad (1.4)$$

where $0 < \beta_1 < \frac{1}{2}$ and $\beta_1 < \beta_2 < 1$.

4. Set $x_{k+1} = x_k + \alpha_k d_k$;
5. Update the matrix H_k

$$H_{k+1} = H_k + \Delta H_k \quad (1.5)$$

where ΔH_k is a low-rank correction matrix. Usually, $H_1 = I$ is chosen, and the updating matrix H_{k+1} is chosen to satisfy the following quasi-Newton equation

$$H_{k+1} y_k = \delta_k \quad (1.6)$$

where $\delta_k = x_{k+1} - x_k, y_k = g_{k+1} - g_k$.

A general class of QN updates was proposed by Broyden [4], given in the form by Shanno [23] can be written as:

$$H^* = H + t \frac{\delta \delta^T}{\delta^T y} + \frac{[(1-t)\delta - Hy][(1-t)\delta - Hy]^T}{[(1-t)\delta - Hy]^T y} \quad (1.7)$$

where t is the scalar parameter, and the subscripts have been omitted by dropping (k) and replacing $(k+1)$ by the subscript $(*)$.

There are three popular choices of t . These are the symmetric rank-one(SR1)update, corresponding to $t = 0$, with

$$H^*(0) = H + \frac{(\delta - Hy)(\delta - Hy)^T}{(\delta - Hy)^T y} \quad (1.8)$$

the DFP update, corresponding to $t = 1$,with

$$H^*(1) = H - \frac{H\delta\delta^T H}{y^T H y} + \frac{\delta\delta^T}{\delta^T y} \quad (1.9)$$

and the BFGS update , corresponding to $t = \infty$,with

$$H^*(\infty) = H - \frac{Hy\delta^T + \delta y^T H}{\delta^T y} + \left(1 + \frac{y^T H y}{\delta^T y}\right) \frac{\delta\delta^T}{\delta^T y} \quad (1.10)$$

The BFGS update has been used successfully in many production codes for solving UNOPs. In practice, it is observed that BFGS out-performed DFP updates in solving practical problems, see [21] for instance. Compared with other QN updates, the SR1 update makes a symmetric rank-one change to the previous approximate inverse Hessian matrix H_k , and thus it is a simpler update to use and requires less computational effort for iteration. A setback with the SR1 update, however , is the fact that its denominator may be zero or nearly zero, which causes numerical instability. Another difficulty is that the SR1 update may not preserve positive definiteness even when H_k is positive definite. However, we observe in practice that when SR1 update solves a given problem, its efficiency is at least, if not better , as good as other QN updates, see [18] for instance.

To improve the performance of QN updates, Biggs [1] proposed to choose H_{k+1} to satisfy the following modified QN equation:

$$H_{k+1}y_k = \lambda_k\delta_k \quad (1.11)$$

where $\lambda_k > 0$ is a scaling parameter.

He showed that a modified BFGS could be derived as follows:

$$H^*(t_k) = H - \frac{Hy\delta^T + \delta y^T H}{\delta^T y} + \left(\frac{1}{t_k} + \frac{y^T H y}{\delta^T y}\right) \frac{\delta\delta^T}{\delta^T y} \quad (1.12)$$

where

$$t_k = \frac{1}{\lambda_k} = \frac{6}{\delta_k^T y_k} (f(x_k) - f(x_{k+1}) + \delta_k^T g_{k+1}) - 2 \quad (1.13)$$

In practice, we observe from our numerical results and results of many other papers that a particular QN algorithm may be "good" in solving certain types of UNOPs, but its efficiency degenerates when it is applied to solve other categories of problems, see [17] for instance. It

is therefore the prime objective of this paper to investigate parallel algorithm based on the relative merits of various QN updates. IN particular, section 2 deals with self-scaling algorithms that are developed in conjunction with QN updates. Self-scaling parallel quasi-Newton(SSPQN) algorithms are proposed in section 3. Implementation issues and numerical results are discussed in section 4. Conclusions remarks and future research directions are also presented in this section.

2 Self-Scaling QN Methods

We propose in this paper another combined QN update formula:

$$H_{k+1} = \left[H_k - \frac{H_k y_k y_k^T H_k}{\delta_k^T H_k y_k} + \theta_k (y_k H_k y_k) v_k v_k^T \right] \gamma_k^{(1)} + \frac{\delta_k \delta_k^T}{t_k \delta_k^T y_k} \quad (2.1)$$

where $\theta_k \in R^1$, t_k is defined in (1.13),

$$v_k = \frac{\delta_k}{\delta_k^T y_k} - \frac{H_k y_k}{y_k^T H_k y_k} \quad (2.2)$$

and $\gamma_k^{(1)}$ is defined in (3.6)

In the following section, we will discuss the motivation to this new QN update.

It is widely accepted that when the inexact linear search is used in conjunction with the QN method, the BFGS method is superior than other quasi-newton updates. However there are some problems for which the SR1 performs better than BFGS see [18].we also observed that for most problems, the results obtained by using BFGS2 update (1.12) are slightly better than those obtained by the original one. This motivate us to search for the next approximate solution along three different directions generated by different updates and leads us to the PQN algorithm [17]. From the implementation results, we can see the PQN algorithm still can not have better performance in some test problems like the power function, the Penalty II function, which motivate us to find another method to modify our algorithm.

In fact, the self-scaling update proposed by Oren (2.3) has such good characteristics. With a self-scaling parameter γ_k the Broyden family can written as

$$H_{k+1} = \left[H_k - \frac{H_k y_k y_k^T H_k}{\delta_k^T H_k y_k} + \theta_k (y_k H_k y_k) v_k v_k^T \right] \gamma_k + \frac{\delta_k \delta_k^T}{\delta_k^T y_k} \quad (2.3)$$

where $\theta_k \in R^1$ and v_k is defined in (2.2). Oren and Luenberger [15] suggest to use the self-adaptable values for the parameter γ_k

$$\gamma_k = t \frac{g_k^T \delta_k}{g_k^T H_k y_k} + (1-t) \frac{\delta_k^T y_k}{y_k^T H_k y_k} \quad (2.4)$$

and usually the value $t = 0$ is recommended for the updates in the convex class.

A lot of numerical experiments on the SSQN methods have been conducted, see, for example,

Table 1: Comparison of BFGS and SSQN methods

Problems	Variables	BFGS	SSQN
	n	Iter/Ifun	Iter/Ifun
Power	20	280/281	28/29
	200	1402/1403	60/62
Penalty II	50	531/543	209/226
Trigonometry	20	44/50	86/92
	200	47/55	104/112

Oren [13, 14], Shanno and Phua [24], Luksan [11]. A research by Shanno and Phua [24] indicates that, for most of the test problems, the Oren and Luenberger’s scaling algorithm actually deteriorates the performance of the BFGS method if the scaling is used at all the iterations. Thus they suggest to scale the approximate Hessian B_k (or approximate inverse Hessian H_k) only at the end of the first iteration, such that a more accurate information on the real Hessian can be incorporated into the updates. This strategy has been widely accepted by the numerical analysts.

Another drawback of SSQN is that it can not guarantee the update matrix sequence $\{B_k\}$ to converge to the real Hessian even if the objective is a quadratic function and the line searches are exact [24]. This property prevent the SSQN from having a quadratic termination property, one of the most important characteristics of a good algorithm.

More recently, Nocedal and Yuan have proved the global convergence of the Oren and Luenberger’s self-scaling QN methods. They also indicate that, for this SSQN methods, it is difficult to estimate the step lengths to ensure superlinear convergence, and thus usually two or more function/gradient evaluations may be required if the superlinear convergence of the methods is required. They have given a counterexample which shows that the step size sequence $\{\alpha_k\}$ has two limit points, 2 and $\frac{1}{2}$, in order to guarantee the superlinear convergence. This property has also been observed in our numerical experiments. This means that this SSQN has destroyed a good property of the original BFGS method, that is, the unit step size is eventually acceptable by the algorithm in inexact linear searches.

However, as Shanno and Phua [24] have observed, the SSQN is indeed superior, for some categories of objective functions, to the original BFGS method (also see, the *refs* in [16]). The most well-known category is the homogeneous function defined by

$$f(x) = \pi^{-1}(x - x^*)^T g(x) + f(x^*) \tag{2.5}$$

where π is the degree of the homogeneous function and x^* is the minimizer. The typical one with degree 4 is the Power's function

$$f_1(x) = (x^T A x)^2, \quad \text{with } A = \text{diag}(1, 2, \dots, n) \quad (2.6)$$

Table 1 outlines the experimental results for this function, Penalty II and the Trigonometric functions by using the BFGS method and the SSQN with the self-scaling parameter defined by Oren and Luenberger

$$\gamma_k = \frac{\delta_k^T y_k}{y_k^T H_k y_k} \quad (2.7)$$

The computation is run on a SUN-Sparc workstation, using FORTRAN-77 in double-precision.

We see from *Table 1* that, for Power and Penalty II functions, SSQN significantly improves the computation efficiency of the original BFGS method. However, for the Trigonometric function, the situation is converse: the SSQN seriously degenerates the efficiency of the BFGS method. Our computational experience is that, for those problems for which the BFGS method performs very badly, the SSQN can generally solve the problems very satisfactorily, such as the Power's functions; conversely, for those problems for which the BFGS performs very well, the SSQN usually needs to take more iterations/function evaluations to find the minimum. Since in practice, we often do not know the properties of the objectives in advance, it is important to design algorithms which are efficient for all problems, such that the performance of the new algorithms is as good as the SSQN methods when applying to the homogeneous functions, and as the BFGS methods when applying to other problems.

Based on the former analysis, we propose our new SSPQN algorithm in the subsequent section.

3 Self-Scaling Parallel quasi-Newton Method (SSPQN)

Our interest here is the parallel extension to the QN methods which will suit for the solution of large-scale optimization problems. The parallelization of the QN methods has been considered by several researchers in the past decades. The earliest work was done for the symmetric rank-one method by Straeter [26] and later was modified by Laarhoven [28]. At each step, this algorithm updates the Quasi-Newton matrix along n independent directions by parallelly evaluating values of the function and the gradient at n 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 in [28] 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 by a factor of 2.5 and the total number of iterations by a factor of 1.5. However, it is assumed that as many processors as needed are available to perform all the computations that can be done in

parallel. This may not be feasible for large problems that arise in many applications. Also, the numerical viability of the method for larger problems remains to be investigated.

A more straightforward approach to parallelize the QN methods is to approximate the derivatives of the function $f(x)$ by using finite difference when the calculation of the exact gradient of the function is prohibitively expensive or is not analytically available. Schnabel [22] suggested to concurrently compute the function values needed by both the gradient approximation and linear search on parallel computers. As the dimensions of the problem grow larger, for a fixed number of processors, this finite difference method will result in a speedup factor that is approaching optimal. Further, in [5], Byrd, *et al.* incorporated the columns of the finite difference Hessian into the BFGS update. Two ways of this incorporation were described and the convergence of the resulting algorithms was established in [5].

Phua [17] proposed another multi-step, multi-directional parallel quasi-Newton methods(PQN) for unconstrained optimization problems . These algorithms generate several QN directions at each iteration, different line search strategies are applied in parallel along each search direction. Numerical experiments are carried out over a wide range of standard test functions. Computational results show that a reduction of 94% and 70% in the number of iterations and function/gradient evaluations respectively could be achieved by the new parallel algorithm. Furthermore, a speedup factor of 1.69 in CPU time could also be realized comparing with serial QN methods. However, one major disadvantage of the PQN algorithms is that it has a poor performance on homogeneous function like Power and Penalty II. This is mainly due to the fact it , like the other broyden class updates, tends to underestimate the eigenvalues of the inverse Hessian matrix.

We add the self-scaling strategy to the former PQN algorithm, then comes our new SSPQN algorithm:

Self-Scaling Parallel Quasi-Newton Algorithm (SSPQN)

1. Initialization

Let $k := 0$, and x_0 be the initial guess of the minimum and $H_0 = I$ be the identity matrix.
Let $\epsilon > 0$ be 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

$$d_k^{(j)} = -H_k(t_j)g_k, \quad j = 1, 2, \dots, m_1 \quad (3.1)$$

For instance, t_j can be chosen as $0, 1, \infty$ in (2.1), t_k in (1.5)-(1.7), or any other appropriate values.

4. Apply the line search(LS) algorithm

Call the LS routine to perform the line search procedure in parallel along each search direction $d_k^{(j)}$, $j = 1, 2, \dots, m_1$. Terminate the execution of this routine once a line minimum α_k has been found satisfying the following Wolfe's conditions:

$$f(x_k + \alpha_k d_k^{(j)}) - f(x_k) \leq 0.0001 \times \alpha_k g_k^T d_k^{(j)} \quad (3.2)$$

and

$$\nabla f(x_k + \alpha_k d_k^{(j)}) \geq 0.9 \times g_k^T d_k^{(j)} \quad (3.3)$$

along any search direction $d_k^{(j)}$. Let d_k^* be the search direction that α_k has been found successfully.

5. Compute the new point

Let

$$\begin{aligned} x_{k+1} &:= x_k + \alpha_k d_k^* \\ g_{k+1} &:= \nabla f(x_{k+1}). \end{aligned}$$

6. Test for convergence

If $\|g_{k+1}\| \leq \epsilon \cdot \max\{1, \|x_{k+1}\|\}$, then stop; Otherwise, proceed to Step 7.

7. Compute the new QN updates

$$H_{k+1} = \left[H_k - \frac{H_k y_k y_k^T H_k}{\delta_k^T H_k y_k} + \theta_k (y_k H_k y_k) v_k v_k^T \right] \gamma_k^{(1)} + \frac{\delta_k \delta_k^T}{t_k \delta_k^T y_k} \quad (3.4)$$

where θ_k, v_k are defined in (2.2) and the value of t_k , as suggested by Biggs, is given by

$$t_k = \frac{6}{\delta_k^T y_k} [f(x_k) - f(x_{k+1}) + \delta_k^T g(x_{k+1})] - 2 \quad (3.5)$$

and $\gamma_k^{(1)}$ is the scaling parameter, which is defined in subsequent subsection. Repeat the whole process from Step 2.

3.1 scaling strategies

- **Scaling Strategy I**

The first scaling strategy is our own contribution, based on our former work on the study of SSQN algorithm's characteristics. It is defined as follows:

$$\gamma_k^{(1)} = \begin{cases} \varepsilon_1, & \text{if } \gamma_k \leq \varepsilon_1 \\ \gamma_k, & \text{if } \varepsilon_1 < \gamma_k \leq \varepsilon_2 \\ \varepsilon_2, & \text{if } \gamma_k > \varepsilon_2 \end{cases} \quad (3.6)$$

where ε_1 is a small positive constant and ε_2 is a bigger constant. γ_k is defined in (2.7).

In practice, we have several options to calculate the self-scaling parameter. One straightforward way is to use the X and G which obtains the line search result with the minimum function value (we call this *optimum scaling*). The other options are what we called *variational scaling strategies*: these include the following strategies :

- SR1 scaling, which uses the X and G of SR1 line search result
- BFGS scaling, which uses the X and G of BFGS line search result
- BFGS2 scaling, which uses the X and G of Biggs' BFGS line search result

- **Scaling Strategy II**

Another strategy called **Controlled Scaling(CS)** strategy is motivated by the work of Luksan [11]. In his work, he give out the scaling strategy based on the results of current line search. The line search algorithm generates a sequence of values $\alpha_1, \alpha_2, \dots, \alpha_p$ such that $\alpha_* = \alpha_p$ is the first value satisfying the wolfe condition (1.3),(3). The main significance of scaling consists in its influence on the requirement of the initial step size acceptability ($\alpha^* = \alpha_1 = 1$ should be satisfied for most iterations). There are two influences against each other: one is at the beginning of the iterative process, the scaling usually adapts the initial matrices to be closer to the corresponding Hessians which increases the probability of $\alpha_1 = 1$ being accepted. The other is at the end of the iterative process, the scaling can violate convergence of the final matrices to the corresponding Hessians, so that $\alpha_1 = 1$ may not be a suitable choice. Thus Luksan proposes a controlled scaling strategy which suggests that scaling should not be used whenever the step size $\alpha_1 = 1$ gives a successful result. We adopt his proposed algorithm in our SSPQN algorithm. The modified CS scaling algorithm is defined as follows:

Given $F, F_1 = F(x + \alpha_1 d), F^*, g, g_1 = g(x + \alpha_1 d), g^*, d$.

- 1 . Compute scaling parameter $\gamma_k > 0$. If in the first iteration, exit.
- 2 . Compute the ratio $\lambda_1 = d^T g_1 / d^T g$. if $\|\lambda_1\| \leq 0.2$ and $F_1 \leq F$, then set $\gamma = 1$.
- 3 . if $\gamma > 1$ and if $F_1 > F$ or $\lambda_1 < 0$, then set $\gamma = 1$.
- 4 . if $\gamma < 1$ and if $F_1 \leq F$ or $\lambda_1 > 0$, then set $\gamma = 1$.
- 5 . if $\gamma < 0.5$ or $\gamma > 2.5$ then set $\gamma = 1$.

Since our SSPQN algorithm utilizes different search directions , we define d in the above strategy with three different search directions, namely SR1, BFGS, Biggs' BFGS.

4 Numerical Results and Discussion

To implement the proposed SSPQN methods, we have chosen three parallel processes ($m_1 = 3$) for computing the following quasi-newton directions concurrently in step 3 of the **SSPQN** routine

$$d_k^{(1)} = -H_k(0)g_k \quad (4.1)$$

$$d_k^{(2)} = -H_k(\infty)g_k \quad (4.2)$$

and

$$d_k^{(3)} = -H_k(t_k)g_k \quad (4.3)$$

That is, three parallel search directions are generated by SR1, BFGS and Biggs' updates defined in (1.8), (1.4) and (1.5)-(1.7) respectively.

In our implementation, the programs are all written in FORTRAN 90 language on top of the PVM(Parallel Virtual Machine), which are run under the CRAY J916 supercomputing environment. We also use the Master-Slave computing model to implement the SSPQN algorithms. The main program acts as the “master”, who controls the three “slaves” . Each “slave” is delegated the task of performing the line search along a particular search direction.

The required accuracy is set as: $\epsilon = 10^{-5}$. That is, convergence is assumed if the following criterion is satisfied at the point x_k

$$\|g_k\| \leq 10^{-5} \times \max\{1, \|x_k\|\} \quad (4.4)$$

where $\|\cdot\|$ is the l_2 (Euclidean) norm. In practice, one may wish to adopt the convergence criterion as proposed by Dennis and Schnabel[7] on the relative gradient of f at $x = x^{(k)}$, that is, when the condition (4.5) defined as follows is satisfied:

$$\max \left| g_i^{(k)} \times \max\{|x_i^{(k)}|, |x_i^*|\} \right| \leq \epsilon \times \max\{|f(x^{(k)})|, f^*\}, 1 \leq i \leq n \quad (4.5)$$

where x^* and f^* are the user's estimate of typical magnitudes of x and f , respectively. The test condition(4.5) is independent of any change in the units of f and x .

A total of 11 standard functions have been chosen for evaluation purposes. Several of these functions are given by Power [13], Broyden-Toint [27] and Hilbert [3], and the remaining functions are described in [12]. Most of these test problems are also available in the MINPACK and CUTE libraries [2]. Each function, except for Wood and Penalty Function II, is tested with dimensions varied from 20 to 1000 variables. Wood's function is tested with $n = 4$, while the Penalty Function II is tested with $n = 20$ and 50, respectively. This represents a total of 63 test problems.

For the purpose of comparison, two serial quasi-Newton methods are also evaluated over the same set of test problems. These include the CONMIN, developed by Shanno and Phua [25] for the BFGS algorithm (1.3) and E04DGE, the routine in the NAG library for solving nonlinear unconstrained problems. Numerical results obtained by SSPQN1 with optimum scaling (OS) strategy, CONMIN and E04DGE algorithms are summarized in Table 2. Table 3 shows the computational results of SSPQN1 with variational scaling (VS) strategies. Table 4 gives the computational results of SSPQN2 with controlled scaling (CS) strategies. The comparison of the best SSPQN results with CONMIN, E04DGE are summarized in table 5.

In these tables, "Iter" denotes the number of iterations and "Ifun" denotes the number of function/gradient evaluations. For SSPQN, whenever two or three function/gradient evaluations can be done concurrently, they are counted as one function/gradient evaluation.

Since CONMIN has failed to solve the Watson function for $n = 1000$, as the maximum allowable CPU time is exceeded, the remaining 56 problems will be considered when the evaluation of SSPQN and CONMIN algorithms is performed. In solving this set of 56 test problems, the total number of iterations and function/gradient evaluations required by SSPQN and CONMIN are 6113 and 6549 versus 20271 and 21615 respectively. Thus, the savings gained by SSPQN in terms of number of iterations and function/gradient evaluations are 14158 and 15066, which represents the speedup factors of 3.32 and 3.30, respectively. The greatest savings are realized by SSPQN for the Power function when $n = 800$ and $n = 1000$, in this case, the speedup factor of up to 25 times is achieved.

As for E04DGE, it has failed to solve the Broyden-Toint function for $n = 1000$ due to the fact that internal errors occurred in some routines of this package. Thus, in comparing the performance of SSPQN and E04DGE, only 56 problems(excluding the above problem) will be considered. For this set of problems, the total number of iterations and function/gradient evaluations requested by SSPQN and E04DGE are 4686 and 5122 versus 17836 and 21091 respectively. Hence the savings gained by SSPQN, in terms of the number of iterations and function/gradient evaluations, are 13150 and 15969 respectively, which represent the speedup factors of 3.81 and 4.12 respectively.

5 Conclusions

We have presented a class of multi-step, multi-directional parallel algorithms for solving unconstrained optimization problems. The proposed algorithms generate several quasi-Newton directions at each iteration and different line search strategies are then applied in parallel along each direction. The numerical results for a broad class of test problems show that a reduction of 200%, or more, in terms of the number of iterations and function/gradient evaluations can be obtained over the serial BFGS methods.

In practice, we also note that as the size and complexity of the problem increase, greater improvements could be realized by our SSPQN algorithms. The numerical results also indicate that the parallel algorithms are efficient and robust in solving large-scale problems. Our numerical results show that the speedup factor of up to 25 times could be achieved by the new parallel quasi-newton algorithm in solving certain large-scale test problems when it is compared with the performance of some serial quasi-newton algorithms.

As our implementation is presently limited to 3 parallel processors, that is, three parallel processors are required to find the line minimum along each of the three search directions simultaneously, further experiments can be carried out to include more parallel processors so that more search directions and line search strategies could be incorporated into the algorithm.

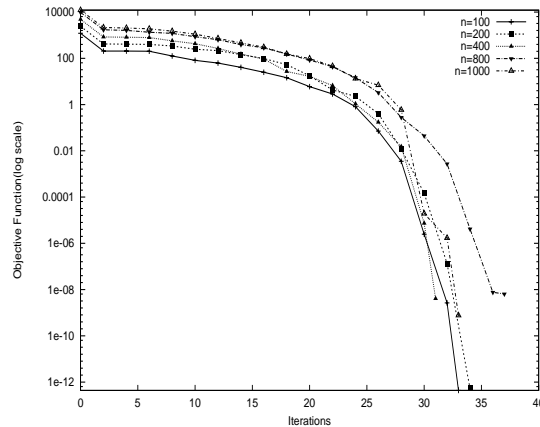


Figure 1: Variations of the objective function of various dimensions for Rosenbrock Function

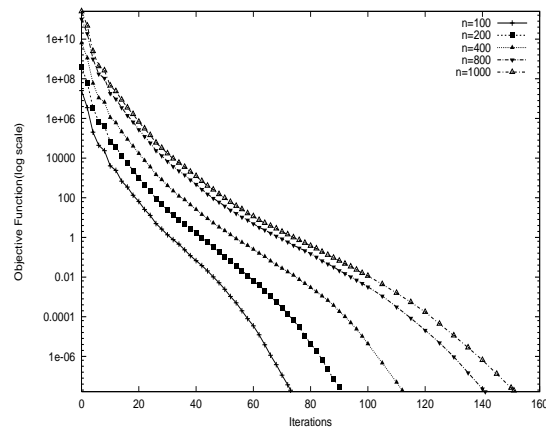


Figure 2: Variations of the objective function of various dimensions for Power Function

References

- [1] M.C. Biggs, A note on minimization algorithms which make use of non-quadratic properties of the objective function, *J. Inst. Maths Applies.* **12** (1973) 337-338.
- [2] I.Bongartz and A. R. Conn, Nick Gould and Ph.L. Toint, CUTE: Constrained and Unconstrained Testing Environment *ACM Transactions on Mathematical Software* **21** (1995) 123-160
- [3] R.P. Brent, *Algorithms for Minimization without Derivatives*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.
- [4] C.G. Broyden, Quasi-Newton methods and their application to function minimization, *Math. Comp.*, **21** (1967) 368-381.
- [5] R.H. Byrd, R.B. Schnabel and G. A. Shultz, Parallel quasi-newton methods for unconstrained optimization, *Mathematical Programming* **42** (1988) 273-306.
- [6] W.C. Davidon, Variable metric method for minimization, *Computer Journal* **10** (1968), 406-410.
- [7] J.E. Dennis, J.R. and R.B. Schnabel, Numerical methods for unconstrained optimization and non-linear equations, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, (1983).
- [8] R. Fletcher, A new approach to variable metric algorithms, *Computer Journal* **13** (1970) 392-399.
- [9] D. Goldfarb, A family of variable metric methods derived by variational means, *Mathematics of Computations* **24** (1970) 23-26.
- [10] F.A. Lootsma and K.M. Ragsdell, State-of-the art in parallel nonlinear optimization, *Parallel Computing* **6** (1988) 133-155.
- [11] L. Luksan, Computational experience with known variable metric updates, *Journal of Optimization Theory and Applications*, **83** (1994) 27-47.
- [12] J.J. More, B.S. Garbow and K.E. Hillstrom, Testing unconstrained optimization software, *ACM Transactions on Mathematical Software* **7** (1981) 17-41.
- [13] S.S. Oren, Self-scaling variable metric algorithms Part II, *Management Science* **20** (1974) 863-874.

- [14] S.S. Oren, On the selection of parameters in self scaling variable metric algorithms, *Math. Prog.* **7** (1974) 351-367
- [15] S.S. Oren and D.G. Luenberger, Self-scaling variable metric (SSVM) algorithms, Part I: Criteria and sufficient conditions for scaling a class of algorithms, *Management Science* **20** (1974) 845-874
- [16] S.S. Oren, Perspectives on self-scaling variable metric algorithms, *JOTA* **37** (1982) 137-147.
- [17] K.H. Phua, Multi-directional parallel algorithms for unconstrained optimization, *Optimization* **38**(1996) 107-125.
- [18] K.H. Phua and S.B. Chew, Symmetric rank-one update and quasi-Newton methods, in: K.H. Phua et al., eds., *Optimization Techniques and Applications*, World Scientific, Singapore (1992) 52-63.
- [19] K.H. Phua, Weiguo Fan, Yuelin Zeng, Parallel algorithms for large-scale nonlinear optimization, *Technical Report TR22/96*, Dept of Information Systems & Computer Science, National University of Singapore, 1996
- [20] K.H. Phua and R. Setiono, Multi-step, multi-directional parallel algorithms for unconstrained optimization (1993), in: K.H. Phua et al., eds., *Optimization Techniques and Applications*, World Scientific, Singapore (1992) 481-487.
- [21] M.J.D Powell, How bad are the BFGS and DFP methods when the objective function is quadratic, *Mathematical Programming*, **34** (1986) 34-47.
- [22] R.B. Schnabel, Concurrent function evaluations in local and global optimization, *Computer Methods in Applied Mechanics and Engineering* **64** (1987) 537-552.
- [23] D.F. Shanno, Conditioning of quasi-Newton methods for function minimization, *Mathematics of Computation* **24** (1970) 647-656.
- [24] D.F. Shanno and P.K.H. Phua, Matrix conditioning and nonlinear optimization, *Mathematical Programming*, vol.14, pp.149-160, 1978
- [25] D.F. Shanno and K.H. Phua, Remark on algorithm 500—a variable method subroutine for unconstrained nonlinear optimization, *ACM Trans. Math. Software* **6** (1980) 618-622.
- [26] T.A. Straeter, A parallel variable metric optimization algorithm. NASA Technical Note, NASA TN D-7329, Langley Research Center, Hampton, VA 23665, 1973.
- [27] Ph.L. Toint, Some numerical results using a sparse matrix updating formula in unconstrained optimization, *Mathematics of Computation* **32** (1985) 68-81.
- [28] P.J.M. van Laarhoven, Parallel variable metric algorithms for unconstrained optimization, *Mathematical Programming* **33** (1985) 68-81.

Table 2. Numerical results obtained by SSPQN1 (OS),
CONMIN and E04DGE programs.

Problems	SSPQN1	CONMIN	E04DGE
	Iter/Ifun	Iter/Ifun	Iter/Ifun
Rosenbrock			
n=20	36/50	34/50	31/42
n=100	34/46	36/50	28/50
n=200	36/54	34/45	28/45
n=400	32/42	34/45	31/53
n=800	33/46	36/45	28/54
n=1000	35/52	35/46	26/48
Powell			
n=20	49/70	51/52	40/44
n=100	42/55	67/68	64/73
n=200	53/65	52/53	53/73
n=400	43/53	69/70	117/182
n=800	57/79	55/58	64/102
n=1000	55/80	49/53	222/364
Power			
n=20	29/30	280/281	100/105
n=100	46/47	867/868	201/225
n=200	64/65	1403/1404	159/186
n=400	87/89	2207/2208	157/191
n=800	118/119	3356/3357	460/473
n=1000	134/135	3857/3858	183/209
Watson			
n=20	87/92	115/119	709/715
n=100	135/143	186/195	1101/1138
n=200	181/189	228/233	1184/1257
n=400	221/238	252/265	1594/1749
n=800	312/331	298/313	2371/2656
n=1000	275/297	F/F *	2304/2678
Broyden-Tridiagonal			
n=20	19/20	21/22	31/40
n=100	21/22	22/23	45/84
n=200	22/23	25/26	67/126
n=400	21/24	28/30	90/176
n=800	22/30	45/47	97/186
n=1000	21/30 15	60/65	98/187
Subtotal:	2320/2616	13802/13949	11683/13511
(30 Problems)		+F/+F	+F/+F

Table 2.(continued) Numerical results obtained by SSPQN1 (OS),
CONMIN and E04DGE programs.

Problems	SSPQN1	CONMIN	E04DGE
	Iter/Ifun	Iter/Ifun	Iter/Ifun
Trigonometry			
n=20	35/43	44/50	62/71
n=100	37/45	42/52	58/72
n=200	36/43	47/55	65/79
n=400	38/46	48/58	56/63
n=800	40/48	58/67	58/72
n=1000	43/51	56/63	60/65
Broyden-Toint			
n=20	26/27	36/37	61/72
n=100	42/43	51/52	50/96
n=200	37/38	47/48	58/117
n=400	46/47	922/926	85/174
n=800	1356/1366	1693/1702	1757/2109
n=1000	1738/1746	2061/2071	F/F **
Wood			
n=4	16/25	23/29	37/44
Hilbert			
n=20	19/22	30/31	64/66
n=100	27/32	46/47	168/172
n=200	30/34	53/55	270/274
n=400	34/38	62/64	272/274
n=800	41/46	71/73	279/285
n=1000	40/45	70/72	344/350
Penalty Function I			
n=20	43/58	62/73	498/729
n=100	40/50	69/78	790/1230
n=200	45/52	59/72	140/147
n=400	47/57	69/80	113/115
n=800	45/51	58/71	220/222
n=1000	52/58	61/73	294/296
Penalty Function II			
n=20	70/88	307/359	181/254
n=50	70/83	523/1051	113/132
Subtotal:	4093/4282	6668/7409	6153/7580
27 Problems			+F/+F
Grand Total:	6413/6898	20470/21358	17836/21091
(57 Problems)		+F/+F	+2F/+2F

* Maximum allowable CPU time is exceeded. 16

** Internal errors occurred.

Tabel 3. Implementation Results of SSPQN1 (VS) on CRAYJ916

Problems		SR1	BFGS	BFGS2
	N	ITER/IFUN	ITER/IFUN	ITER/IFUN
Rosenbroch				
	n=20	54/83	34/44	39/61
	n=100	36/51	33/42	32/40
	n=200	34/42	37/52	34/47
	n=400	46/69	38/51	33/49
	n=800	34/45	39/58	34/43
	n=1000	35/44	33/46	40/65
Powell				
	n=20	35/36	39/40	33/35
	n=100	36/38	35/36	44/50
	n=200	44/47	36/37	42/45
	n=400	46/51	35/37	43/45
	n=800	42/47	37/40	36/40
	n=1000	30/35	35/38	37/40
Power				
	n=20	58/59	93/96	41/42
	n=100	84/87	216/219	73/74
	n=200	102/103	356/358	90/91
	n=400	139/141	506/509	112/113
	n=800	163/167	1100/1102	141/142
	n=1000	164/171	1338/1341	153/154
Watson				
	n=20	65/73	59/64	86/91
	n=100	106/127	127/131	142/149
	n=200	187/221	168/173	159/176
	n=400	247/327	195/206	223/231
	n=800	374/458	216/241	238/257
	n=1000	316/423	225/254	263/284
Broyden-Tridiagonal				
	n=20	21/23	19/20	19/20
	n=100	21/23	21/22	21/22
	n=200	22/23	22/23	22/23
	n=400	21/24	20/37	20/37
	n=800	22/28	20/38	20/38
	n=1000	23/31	20/38	21/40
Subtotal:		2607/3097	5152/5393	2291/2544
(30 Problems)				

Table 3(continued). Implementation Results of SSPQN1 (VS) on CRAYJ916

Problems	N	SR1	BFGS	BFGS2
		ITER/IFUN	ITER/IFUN	ITER/IFUN
Trigonometry				
	n=20	35/44	35/40	37/48
	n=100	40/61	38/47	37/45
	n=200	34/42	42/54	34/40
	n=400	40/51	39/46	38/45
	n=800	41/51	41/51	41/49
	n=1000	38/45	38/45	42/50
Broyden-Toint				
	n=20	28/31	28/29	28/29
	n=100	40/41	38/39	41/42
	n=200	39/42	36/37	38/39
	n=400	44/52	44/45	44/45
	n=800	1126/1449	1319/1326	1335/1349
	n=1000	1454/1552	1698/1745	1690/1711
Wood				
	n=4	18/28	18/26	15/24
Hilbert				
	n=20	16/19	14/16	17/20
	n=100	21/26	25/26	30/31
	n=200	26/30	28/32	29/32
	n=400	25/33	26/29	28/31
	n=800	35/39	31/34	41/45
	n=1000	39/49	41/47	44/48
Penalty Function I				
	n=20	31/39	43/53	46/57
	n=100	40/52	45/51	39/49
	n=200	49/61	41/50	51/60
	n=400	51/59	53/58	43/55
	n=800	53/63	55/62	58/64
	n=1000	51/66	47/58	44/53
Penalty Function II				
	n=20	130/169	101/134	95/117
	n=50	117/140	181/194	100/111
Subtotal:		3661/4334	4145/4374	4085/4289
27 Problems				
Grand Total:		6268/7431	9297/9767	6376/6833
(57 Problems)				

Table 4. Implementation Results of SSPQN2 (CS) on CRAYJ916

Problems		SR1	BFGS	BFGS2
	N	ITER/IFUN	ITER/IFUN	ITER/IFUN
Rosenbroch				
	n=20	30/39	32/42	34/44
	n=100	30 /39	33/45	33/45
	n=200	32/39	34/40	36/42
	n=400	32/40	31/36	31/38
	n=800	32/41	43/63	43/63
	n=1000	30/38	35/46	30/36
Powell				
	n=20	39/41	38/40	39/41
	n=100	36/41	44/48	44/48
	n=200	41/45	45/52	35/38
	n=400	48/50	40/42	49/53
	n=800	49/55	42/47	42/47
	n=1000	37/45	38/46	38/46
Power				
	n=20	41/42	41/42	41/42
	n=100	73/74	73/74	73/74
	n=200	93/94	90/91	90/91
	n=400	114/115	112/113	112/113
	n=800	143/144	141/142	141/142
	n=1000	160/161	151/152	151/152
Watson				
	n=20	89/96	86/91	86/91
	n=100	154/161	163/173	163/173
	n=200	189/197	189/197	189/196
	n=400	230/243	222/238	222/236
	n=800	277/287	328/364	328/364
	n=1000	289/306	291/305	284/296
Broyden-Tridiagonal				
	n=20	19/20	19/20	19/20
	n=100	21/22	21/22	21/22
	n=200	22/23	22/23	22/23
	n=400	21/24	21/24	21/24
	n=800	22/28	23/27	22/26
	n=1000	23/31	23/27	22/26
Subtotal:		2416/2581	2471/2672	2462/2655
(30 Problems)				

Table 4(continued) Implementation Results of SSPQN2 (CS) on
CRAYJ916

Problems		SR1	BFGS	BFGS2
	N	ITER/IFUN	ITER/IFUN	ITER/IFUN
Trigonometry				
	n=20	37/44	38/44	37/44
	n=100	38/47	37/45	37/45
	n=200	42/51	35/42	35/42
	n=400	39/49	40/49	40/49
	n=800	42/51	41/54	40/47
	n=1000	43/50	40/47	41/50
Broyden-Toint				
	n=20	27/28	27/28	27/28
	n=100	41/42	41/42	41/42
	n=200	37/38	37/38	37/38
	n=400	45/46	45/46	45/46
	n=800	1358/1362	1358/1368	1358/1368
	n=1000	1672/1691	1684/1696	1684/1696
Wood				
	n=4	16/25	21/30	21/30
Hilbert				
	n=20	9/13	10/14	10/14
	n=100	23/27	29/33	29/33
	n=200	26/32	31/35	31/35
	n=400	45/48	32/37	32/37
	n=800	39/42	43/48	43/48
	n=1000	46/51	45/49	45/49
Penalty Function I				
	n=20	44/51	40/50	43/52
	n=100	50/59	51/60	52/62
	n=200	40/52	43/55	53/63
	n=400	50/59	47/55	52/62
	n=800	56/63	58/65	54/63
	n=1000	61/68	61/70	60/70
Penalty Function II				
	n=20	117/140	106/138	92/112
	n=50	102/106	86/94	111/120
Subtotal:		4135/4335	4126/4332	4150/4345
27 Problems				
Grand Total:		6561/6916	6597/7004	6612/7000
(57 Problems)				

Table 5. Comparison of results obtained by CONMIN, E04DGE with respect to SSPQN's best results

Problems	Difference2.1	Difference2.2	Speedup2.1	Speedup2.2
	CONMIN	E04DGE	CONMIN	E04DGE
	Iter/Ifun	Iter/Ifun	Iter/Ifun	Iter/Ifun
Rosenbrock				
n=20	-5/-11	-8/-19	0.88/0.82	0.80/0.69
n=100	4/10	-4/10	1.13/1.25	0.88/1.25
n=200	0/-2	-6/-2	1.00/0.96	0.83/0.96
n=400	1/-4	-2/4	1.04/0.92	0.94/1.09
n=800	2/2	-6/11	1.06/1.05	0.83/1.26
n=1000	-5/-19	-14/-17	0.88/0.71	0.65/0.74
Powell				
n=20	18/17	7/9	1.55/1.49	1.22/1.26
n=100	23/18	20/23	1.53/1.36	1.46/1.46
n=200	10/8	11/28	1.24/1.18	1.27/1.63
n=400	26/25	74/137	1.61/1.56	2.73/4.05
n=800	19/18	28/62	1.53/1.45	1.78/2.55
n=1000	12/13	185/324	1.33/1.33	6.00/9.10
Power				
n=20	239/239	59/63	6.83/6.70	2.44/2.50
n=100	794/794	128/151	11.88/11.73	2.76/3.05
n=200	1313/1313	69/95	15.59/15.43	1.77/2.05
n=400	2095/2095	45/78	19.71/19.54	1.41/1.70
n=800	3215/3215	319/331	23.81/23.65	3.27/3.34
n=1000	3704/3704	30/55	25.21/25.06	1.20/1.36
Watson				
n=20	29/28	623/624	1.34/1.31	8.25/7.86
n=100	44/46	959/989	1.31/1.31	7.76/7.64
n=200	69/57	1025/1081	1.44/1.33	7.45/7.15
n=400	29/34	1371/1518	1.14/1.15	7.15/7.58
n=800	60/56	2133/2399	1.26/1.22	9.97/10.34
n=1000	F/F*	2041/2394	F/F*	8.77/9.43
Broyden-Tridiagonal				
n=20	2/2	12/20	1.11/1.10	1.64/2.00
n=100	1/1	24/62	1.05/1.05	2.15/3.82
n=200	3/3	45/103	1.14/1.14	3.05/5.48
n=400	8/-7	70/139	1.40/0.82	4.50/4.76
n=800	25/9	77/148	2.25/1.24	4.85/4.90
n=1000	39/25	77/147	2.86/1.63	4.67/4.68
Subtotal: (30 Problems)	11511/11405	9392/10967	6.03/5.49	5.10/5.32

Table 5(continued). Comparison of results obtained by CONMIN, E04DGE with respect to SSPQN's best results

Problems	Difference2.1	Difference2.2	Speedup2.1	Speedup2.2
	CONMIN	E04DGE	CONMIN	E04DGE
	Iter/Ifun	Iter/Ifun	Iter/Ifun	Iter/Ifun
Trigonometry				
n=20	7/2	25/23	1.19/1.05	1.68/1.48
n=100	5/7	21/27	1.14/1.16	1.57/1.60
n=200	13/15	31/39	1.39/1.38	1.92/1.98
n=400	10/13	18/18	1.27/1.29	1.48/1.40
n=800	17/18	17/23	1.42/1.37	1.42/1.47
n=1000	14/13	18/15	1.34/1.26	1.43/1.30
Broyden-Toint				
n=20	8/8	33/43	1.29/1.28	2.18/2.49
n=100	10/10	9/54	1.25/1.24	1.22/2.29
n=200	9/20	20/78	1.24/1.24	1.53/3.00
n=400	878/881	41/129	20.96/20.58	1.94/3.87
n=800	358/353	422/760	1.27/1.27	1.32/1.57
n=1000	371/360	F/F**	1.22/1.22	F/F**
Wood				
n=4	8/5	22/20	1.54/1.21	2.47/1.84
Hilbert				
n=20	13/11	47/46	1.77/1.55	3.77/3.30
n=100	16/16	138/141	1.54/1.52	5.60/5.55
n=200	24/23	241/242	1.83/1.72	9.32/8.57
n=400	34/33	244/243	2.22/2.07	9.72/8.84
n=800	30/28	238/240	1.74/1.63	6.81/6.34
n=1000	26/24	300/302	1.60/1.50	7.82/7.30
Penalty Function I				
n=20	16/16	452/672	1.35/1.29	10.83/12.79
n=100	30/29	751/1181	1.77/1.60	20.26/25.11
n=200	8/12	89/87	1.16/1.20	2.75/2.45
n=400	26/25	70/60	1.61/1.46	2.63/2.10
n=800	0/7	162/158	1.00/1.11	3.80/3.47
n=1000	17/20	250/243	1.39/1.38	6.69/5.59
Penalty Function II				
n=20	212/242	86/137	3.24/3.07	1.91/2.18
n=50	423/940	13/21	5.23/9.47	1.13/1.19
Subtotal:	2583/3120	2068/3291	1.64/1.73	1.51/1.77
27 Problems				
Grand Total:	14094/14525	11460/14258	3.22/3.13	2.80/3.09
(57 Problems)				