

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, fanwg, 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 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 obtained by the new SSPQN algorithms over the conventional quasi-Newton methods are more than 300%, both 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 times. In general, it is noted that as the size and complexity of the problem increase, greater improvements and savings could be realized by our new parallel algorithms.

1. Introduction

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

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

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 conditions are satisfied:

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

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

Submitted to the Fourth International Conference on Optimization: Techniques and Applications, Australia

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 [1]:

$$H_{k+1}(\phi_k) = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{\delta_k \delta_k^T}{\delta_k^T y_k} + \phi_k (y_k^T H_k y_k) v_k v_k^T \quad (1.7)$$

$$v_k = \frac{\delta_k}{\delta_k^T y_k} - \frac{H_k y_k}{y_k^T H_k y_k}, \quad \beta_k = \frac{y_k^T H_k y_k}{\delta_k^T y_k}, \quad \gamma_k = \frac{\delta_k^T B_k \delta_k}{\delta_k^T y_k} \quad (1.8)$$

$$\phi_k = \phi_k(\theta_k) = (1 - \theta_k)/(1 + \theta_k(\beta_k \gamma_k - 1)) \quad (1.9)$$

where $\theta_k \in R^1$ is a parameter. There are three popular choices of θ_k . These are the symmetric rank-one(SR1)update, corresponding to $\theta_k = \frac{\delta_k^T y_k}{\delta_k^T y_k - \delta_k^T B_k \delta_k}$, the DFP update, corresponding to $\theta_k = 1$ and the BFGS update, corresponding to $\theta_k = 0$. 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 Powell [3] 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 per 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 Phua [4] for instance.

To improve the performance of QN updates, Biggs [5] proposed to choose H_{k+1} to satisfy the following modified QN equation:

$$H_{k+1} y_k = \lambda_k \delta_k \quad (1.10)$$

where $\lambda_k > 0$ is a scaling parameter. He showed that a modified BFGS could be derived as follows:

$$H_{k+1}(\tau_k) = H_k - \frac{H_k y_k \delta_k^T + \delta_k y_k^T H_k}{\delta_k^T y_k} + \left(\frac{1}{\tau_k} + \frac{y_k^T H_k y_k}{\delta_k^T y_k} \right) \frac{\delta_k \delta_k^T}{\delta_k^T y_k} \quad (1.11)$$

where

$$\tau_k = \frac{1}{\lambda_k} = \frac{6}{\delta_k^T y_k} (f(x_k) - f(x_{k+1}) + \delta_k^T g_{k+1}) - 2 \quad (1.12)$$

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 Phua [6] for instance. It is therefore the prime objective of this paper to develop parallel algorithms based on the relative merits of various QN updates.

2. A New Class of Self-Scaling QN Updates

It is widely accepted that when the inexact linear search is used in conjunction with the QN method, the BFGS method is superior to other quasi-Newton updates. However there are some problems for which the SR1 update performs better than the BFGS (Phua [4]). We also observed that for most problems, the results obtained by using BFGS2 update (1.11) are slightly better than those obtained by the original BFGS update. This motivates us to search for the better approximate solution along different directions generated by different updates. Some of these parallel algorithms are considered in Phua [6]. From the implementation results, we can see that the performance of these parallel algorithms can still be further improved.

In fact, the self-scaling update proposed by Oren [7] has some good characteristics. With a self-scaling parameter γ_k , this class of updates can be written as

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

Oren and Luenberger [8] suggest to use the self-adaptable values for the parameter γ_k

$$\gamma_k = t \frac{g_k^T \delta_k}{g_k^T H_k y_k} + (1 - t) \frac{\delta_k^T y_k}{y_k^T H_k y_k} \quad (2.2)$$

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, Oren [7, 9], Shanno and Phua [10], Luksan [11]. A research by Shanno and Phua [10] 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 algorithms is that it can not guarantee the sequence of updates $\{B_k\}$ to converge to the real Hessian even if the objective function is a quadratic function and the line searches are exact [10]. This property prevents SSQN algorithms from having a quadratic termination property, one of the most important characteristics of a good algorithm.

However, as Shanno [10] has observed, SSQN algorithms are indeed superior, for some categories of objective functions, to the original BFGS method (also see Oren [12]). The most well-known category of problems is the homogeneous function defined by

$$f(x) = \pi^{-1} (x - x^*)^T g(x) + f(x^*) \quad (2.3)$$

where π is the degree of the homogeneous function and x^* is the minimizer. Their computational experience shows that, for those problems for which the BFGS method performs very badly, SSQN algorithms can generally solve the problems very satisfactorily, such as the Power functions; while for those problems for which the BFGS method performs very well, SSQN algorithms usually need to take more iterations/function evaluations to find the minimum. Since in practice, we often do not know the properties of the objective functions in advance, it is important to design algorithms which are efficient for all problems.

Having considered the pros and cons of various QN updates, it is our objective now to propose a new class of three-parameter updates which will combine the relative merits of different types of QN updates. This class of modified QN updates can be expressed as follows:

$$H_{k+1}(\phi_k, \gamma_k^{(1)}, \tau_k) = \left[H_k - \frac{H_k y_k y_k^T H_k}{\delta_k^T H_k y_k} + \phi_k (y_k H_k y_k) v_k v_k^T \right] \gamma_k^{(1)} + \frac{\delta_k \delta_k^T}{\tau_k \delta_k^T y_k} \quad (2.4)$$

where ϕ_k is defined by (1.9), τ_k is defined by (1.12), and $\gamma_k^{(1)}$ is a scaling parameter which will be defined at a later stage. It is evident that the class of updates given by (2.4) includes all classes of QN updates that we have mentioned above as special cases. For instance, we have:

- (i) $H_{k+1}(\phi_k, 1, 1) =$ the Broyden family of QN updates
- (ii) $H_{k+1}(\phi_k(0), 1, \tau_k) =$ the Biggs family of modified BFGS updates
- (iii) $H_{k+1}(\phi_k, \gamma_k, 1) =$ the Oren family of self-scaling QN updates

In the next section, parallel QN algorithms will be developed by using this class of updates for the purpose of generating simultaneous parallel search directions at the beginning of each iteration.

3. Self-Scaling Parallel Quasi-Newton Methods (SSPQN)

Phua [6, 13, 14] propose a class of multi-step, multi-directional parallel quasi-Newton methods (PQN) for solving unconstrained optimization problems. These algorithms generate several QN directions at each iteration, different line search strategies are then 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 terms of the total number of iterations and function/gradient evaluations respectively could be achieved by their algorithms. To further improve the practical performance of parallel algorithms, we propose in this paper that the self-scaling quasi-Newton (SSQN) updates defined in (2.4) are used to generate parallel search directions at each iteration. Furthermore, the value of scaling parameter $\gamma_k^{(1)}$ will be chosen appropriately so that the new parallel algorithms will be efficient and robust in solving all types of practical problems. With these considerations in mind, we shall now propose our new parallel algorithms as follows:

Self-Scaling Parallel Quasi-Newton Algorithms (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 parallel processors available for computing the search directions simultaneously. Compute, in parallel,

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

Here the updating matrices $H_k(\phi_{kj}, \gamma_{kj}^{(1)}, \tau_{kj})$ are defined by (2.4).

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

Apply the equation (2.4) to compute in parallel the updating matrices: $H_{k+1}(\phi_k, \gamma_k^{(1)}, \tau_k)$ where ϕ_k is defined by (1.9), τ_k is defined by (1.12), and $\gamma_k^{(1)}$ will be chosen as follows:

$$\gamma_k^{(1)} = \begin{cases} \varepsilon_1, & \text{if } \gamma_k \leq \varepsilon_1 \\ \gamma_k, & \text{if } \varepsilon_1 < \gamma_k \leq \varepsilon_2 \\ \varepsilon_2, & \text{if } \gamma_k > \varepsilon_2 \end{cases} \quad (3.4)$$

where ε_1 is a small positive constant, ε_2 is a bigger constant and γ_k is defined in (2.2). Repeat the whole process from Step 2.

4. Numerical Results and Discussion

The SSPQN algorithms are implemented in FORTRAN 90 on top of PVM in the CRAY J916 PVP supercomputer. Three search directions have been chosen: SR1, BFGS and Biggs' modified BFGS. The required accuracy is set as: $\epsilon = 10^{-5}$. A total of 11 standard functions have been chosen for evaluation purposes. 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 57 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 package, developed by Shanno and Phua [15] for the BFGS algorithm and E04DGE, the optimization code available in the NAG library. Numerical results obtained by SSPQN, CONMIN and E04DGE algorithms are summarized in Table 1. In this table, "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 6138 and 6601 versus 20470 and 21358 respectively. Thus, the savings gained by SSPQN in terms of number of iterations and function/gradient evaluations are 14332 and 14757, which represents the speedup factors of 3.33 and 3.24, 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 4675 and 5152 versus 17836 and 21091 respectively. Hence the savings gained by SSPQN, in terms of the number of iterations and function/gradient evaluations, are 13161 and 15939 respectively, which represent the speedup factors of 3.82 and 4.09 respectively.

5. Conclusions

We have presented a class of self-scaling multi-directional parallel algorithms for solving unconstrained optimization problems. The proposed algorithms generate several quasi-Newton directions at each iteration and the line search strategies are then applied in parallel along each direction. The numerical results for a broad class of test problems show that the parallel algorithms are efficient and robust in solving large-scale problems. In practice, we also note that as the size and complexity of the problem increase, greater improvements could be realized by our SSPQN algorithms.

References

- [1] C.G. Broyden, Quasi-Newton methods and their application to function minimization, *Math. Comp.*, **21** (1967) 368-381.
- [2] D.F. Shanno, Conditioning of quasi-Newton methods for function minimization, *Mathematics of Computation* **24** (1970) 647-656.
- [3] M.J.D Powell, How bad are the BFGS and DFP methods when the objective function is quadratic, *Mathematical Programming*, **34** (1986) 34-47.
- [4] 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.
- [5] M.C. Biggs, A note on minimization algorithms which make use of non-quadratic properties of the objective function, *J. Inst. Maths Applics.* **12** (1973) 337-338.
- [6] K.H. Phua, Multi-directional parallel algorithms for unconstrained optimization, *Optimization* **38**(1996) 107-125.
- [7] S.S. Oren, Self-scaling variable metric algorithms Part II, *Management Science* **20** (1974) 863-874.
- [8] 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.
- [9] S.S. Oren, On the selection of parameters in self scaling variable metric algorithms, *Math. Prog.* **7** (1974) 351-367.
- [10] D.F. Shanno and K.H. Phua, Matrix conditioning and nonlinear optimization, *Mathematical Programming*, vol.14, pp.149-160, 1978.
- [11] L. Luksan, Computational experience with known variable metric updates, *Journal of Optimization Theory and Applications*, **83** (1994) 27-47.
- [12] S.S. Oren, Perspectives on self-scaling variable metric algorithms, *JOTA* **37** (1982) 137-147.
- [13] K.H. Phua, W. Fan, Y. Zeng, Parallel algorithms for large-scale nonlinear optimization, to appear on *the Journal of International Transactions in Operational Research*.
- [14] 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.
- [15] 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.

Table 1. Numerical results obtained by SSPQN,
CONMIN and E04DGE programs.

Problems	SSPQN Iter/Ifun	CONMIN Iter/Ifun	E04DGE 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	60/65	98/187
Subtotal: (30 Problems)	2320/2616	13802/13949 +F/+F	11683/13511

Table 1.(continued) Numerical results obtained by SSPQN, CONMIN and E04DGE programs.

Problems	SSPQN Iter/Ifun	CONMIN Iter/Ifun	E04DGE 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: (57 Problems)	6413/6898	20470/21358 +F/+F	17836/21091 +F/+F

* Maximum allowable CPU time is exceeded.

** Internal errors occurred.