Memory-less SR1 and memory-less BFGS methods for unconstrained optimization

Neculai Andrei
Center for Advanced Modeling and Optimization,
Academy of Romanian Scientists,
54, Splaiul Independenţei, Sector 5,
Bucharest, ROMANIA
E-mail: neculaiandrei70@gmail.com

Technical Report 7/2021
May 18, 2021

Abstract. The memory-less SR1 and the memory-less BFGS methods are presented together with their numerical performances for solving a collection of 800 unconstrained optimization problems with the number of variables in the range [100, 1000]. The convergence of the memory-less SR1 method is proved under the classical assumptions. Comparison between the memory-less SR1 and the memory-less BFGS method show that memory-less SR1 is more robust than the memory-less BFGS. Comparison between memory-less SR1 and BFGS in implementation of Shanno and Phua from CONMIN show that memory-less SR1 method is more efficient and more robust than BFGS method from CONMIN, one of the best implementation of BFGS.

Keywords: Symmetric-rank one SR1; Quasi-Newton BFGS; Wolfe line search; Numerical experiments; Dolan and Moré performance profile

Mathematics Subject Classification: 90C30, 90C53, 90-08

1 Introduction

For solving the minimizing problem

$$\min f(x), \tag{1}$$

where $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function, bounded from below, one of the most effective methods is the quasi-Newton methods. In these methods the search direction d_{k+1} is computed as solution of the following linear algebraic system

$$B_{k+1} d_{k+1} = -g_{k+1}, \tag{2}$$

where B_{k+1} is an approximation to the Hessian $\nabla^2 f(x_{k+1})$ and g_{k+1} is the gradient $\nabla f(x_{k+1})$, or directly as

$$d_{k+1} = -H_{k+1} g_{k+1}, \tag{3}$$

where H_{k+1} is an approximation to the inverse Hessian, i.e. $H_{k+1} = B_{k+1}^{-1}$.
Starting with an initial point x_0 the next approximation x_{k+1} to x^*, solution of (1), is computed as

$$x_{k+1} = x_k + \alpha_k d_k, \tag{4}$$

where α_k is the stepsize often computed by the Wolfe line search [1969, 1971]:

$$f(x_k + \alpha_k d_k) < f(x_k) + \rho \alpha_k g_k^T d_k, \tag{5a}$$
$$\nabla f(x_k + \alpha_k d_k)^T d_k > \sigma \nabla f(x_k)^T d_k, \tag{5b}$$

where $0 < \rho < \sigma < 1$ are some parameters.
These algorithms are efficient and robust for minimizing functions that satisfy certain assumptions and have a super-linear rate of local convergence. Currently, many variants of the updating formula for the approximation to the Hessian (or to the inverse Hessian) are known: symmetric rank-one (SR1) [Broyden, 1967; Davidon, 1959, 1968; Fiacco and McCormick, 1968; Wolfe, 1969, 1971] and the rank-two such as the Davidon-Fletcher-Powell (DFP) update [Davidon, 1991; Fletcher and Powell, 1963], Powell-symmetric-Broyden (PSB) [Powell, 1970] and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update [Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970]. A unifying framework for many of these updates, including both rank-one and rank-two updates was given by Huang [1970]. These updates have been very intensive studied and in general it is accepted that the BFGS method is the most efficient and robust for solving minimizing problems with differential functions. However, the main drawback of BFGS is that it is limited for solving small and medium minimizing functions. It requires a large amount of memory and therefore it involves a large amount of numerical operations.
The purpose of the paper is to introduce the memory-less quasi-Newton methods (memory-less SR1 and the memory-less BFGS) as a technique for solving large-scale minimizing problems and to show their performances for solving large-scale unconstrained optimization problems. The structure of the paper is as follows. Section 2 describes the SR1 method. In Section 3 the memory-less SR1 method is introduced. Section 4 is dedicated to the memory-less BFGS method. The memory-less SR1 and the memory-less BFGS algorithms are presented in Section 5. The convergence of the memory-less SR1 method is described in Section 6. The global convergence is proved under classical assumptions. The numerical results and comparisons among the algorithms are detailed in Section 7. For solving a collection of 800 unconstrained optimization problems up to 10000 variables we have the computational evidence that the memory-less SR1 method is more robust than the memory-less BFGS method. Comparing memory-less SR1 versus the BFGS from CONMIN [Shanno and Phua, 1976; Shanno, 1980] shows that, for solving problems up to 10000 variables, memory-less SR1 is way more efficient and more robust. This section also presents the performances of

these algorithms for solving five applications from MINPACK-2 collection, each of them with 40,000 variables. The conclusion of this study is that the memory-less technique applied here for the SR1 and for the BFGS methods is a simple and efficient technique for adapting these quasi-Newton methods for solving large-scale problems. Besides, it seems that the accuracy of the Hessian approximations along the iterations in quasi-Newton methods is not as crucial in these methods as it is believed.

2 SR1 Method

In the quasi-Newton methods the basic requirement for the updating formula to the Hessian is the so called the secant equation, to be satisfied at each iteration, namely

$$B_{k+1}s_k = y_k \quad \text{or} \quad H_{k+1}y_k = s_k, \tag{6}$$

where $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$.

The symmetric rank-one SR1 update formula, in which we are interested in this paper, can be derived as solution of the following simple problem. "*Given a symmetric matrix B_k and the vectors s_k and y_k, finds a new symmetric matrix B_{k+1} such that $B_{k+1} - B_k$ has rank one and such that the secant equation $B_{k+1}s_k = y_k$ is satisfied*." It is easy to see that if $(y_k - B_k s_k)^T s_k \neq 0,$ then the unique solution of the above problem is

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}. \tag{7}$$

If $y_k = B_k s_k$ then the solution is $B_{k+1} = B_k$. However, if $(y_k - B_k s_k)^T s_k = 0$ and $y_k \neq B_k s_k$, then there is no solution to the problem.
Let H_k be the inverse approximation to the Hessian at iteration k. By using the Sherman-Morrison-Woodbury formula in (7), the following update to the inverse Hessian for SR1 is

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k}. \tag{8}$$

This variant of the algorithm is only applicable in cases in which the inverse H_k exists.

The main drawbacks of SR1 update are as follows.
1) The denominator $(y_k - B_k s_k)^T s_k$ of the SR1 update term in (7) may vanish, i.e. $(y_k - B_k s_k)^T s_k \cong 0$, cases in which B_{k+1} is not well-defined.
2) The step directions computed by using the SR1 updating formula given by (7) may no longer be uniform linear independent, thus leading to slow down the convergence or even the stalling.
3) The SR1 Hessian approximation may not be positive definite along the iterations, thus resulting a direction that does not produce descent.

To prevent the method from failing due to the first drawback one simple remedy is to set $B_{k+1} = B_k$. However, this will slow down the convergence of the method. Conn, Gould and Toint (1991) and Khalfan, Byrd and Schnabel (1993) showed that the denominator of (7) vanishes rarely in practice and setting $B_{k+1} = B_k$ does not have a significant impact on the performances of the SR1 method subject to the number of the iterations or runtimes.
The second drawback is more delicate, being in close connection with the uniform linear independence of the search directions generated by the SR1 algorithm. A more precise definition of the uniform linear independence was given by Conn, Gould and Toint (1991). "A sequence $\{s_k\}$ is uniformly linearly independent if there exist $\xi > 0,$ k_0 and $m \geq n$ such that, for each $k \geq k_0,$ there is n distinct indices $k \leq k_1 \leq k_2 \leq \ldots \leq k_n \leq k+m$ for which the minimum singular value of the matrix $S = \left[\frac{s_{k_1}}{\|s_{k_1}\|}, \cdots, \frac{s_{k_n}}{\|s_{k_n}\|} \right]$ is at least ξ." Conn, Gould and Toint (1991) proved that the sequence of matrices generated by the SR1 formula converges to the exact Hessian, when the sequence of iterates converges to a limit point and the sequence of steps is uniformly linearly independent. Kelley and Sachs [1998] provide similar convergence results removing the first of these assumptions. Fiacco and McCormick [1968] showed that if the search directions are linearly independent and the denominator of (7) is always non-zero, then the SR1 method without line searches minimize a strongly convex quadratic function in at most $n+1$ steps. In this case B_{n+1} is exactly the Hessian of the quadratic function. Observe that this result is significant since it does not require exact line search, as is the case for the BFGS update. Generally, the above condition given by the definition of the uniform linear independency is not implemented in practice, it serves only as one of the main assumptions of a proof that the SR1 approximations to the Hessian converge to the true Hessian as the iterates converge to the solution of (1).
Subject to the uniform linear independency of the search directions Khalfan, Byrd and Schnabel [1993] showed that many problems do not satisfy this requirement, but they proved the local convergence of the SR1 method using only the positive definiteness and boundedness assumptions for the approximate Hessian. More than this Conn, Gould and Toint [1991] proved that if the minimizing function f is twice continuously differentiable and its Hessian is bounded and Lipschitz continuous and the iterates generated by the SR1 method converge to a point x^* and in addition for all k,

$$\left|(y_k - B_k s_k)^T s_k\right| \geq \xi \|y_k - B_k s_k\| \|s_k\|, \tag{9}$$

for some $\xi \in (0,1),$ and the steps s_k are uniformly linearly independent, then

$$\lim_{k \to \infty} \left\| B_k - \nabla^2 f(x^*) \right\| = 0. \tag{10}$$

Often condition (9) is used in implementations of the SR1 method in order to ensure that this update is well behaved. If this condition is not satisfied, then the update is skipped. Conn, Gould and Toint [1991] and Khalfan, Byrd and Schnabel [1993] provide theoretical and computational results, respectively, that if the uniform linear independence assumption is satisfied, then the approximations to the Hessian generated by the SR1 method are more accurate than those generated by BFGS, and SR1 converge faster to the true Hessian than BFGS. Therefore, if all these above drawbacks are addressed in a reliable and efficient manner, then SR1 can be used for solving (1) instead of the rank-two updates. More details on SR1 method concerning the undefined updates, choosing the initial approximate B_0, uniform linear independence of the steps, are found in [Benson and Shanno, 2018] and in [Chen, Lam and Chan, 2019].

3 Memory-less SR1 Method

The *memory-less SR1 method with direct approximation to the Hessian* is obtained by considering $B_k = I$ in (7), i.e.

$$B_{k+1} = I + \frac{(y_k - s_k)(y_k - s_k)^T}{(y_k - s_k)^T s_k}. \tag{11}$$

Observe that this is a very simple formula in which the information about the Hessian is not accumulated from iteration to iteration. Besides, we see that the memory-less SR1 method has the same drawbacks as the SR1 method, i.e. when the denominator $(y_k - s_k)^T s_k$, is zero, or is very close to zero, then the SR1 method is not defined. By considering in (8) $H_k = I$, i.e.

$$H_{k+1} = I + \frac{(s_k - y_k)(s_k - y_k)^T}{(s_k - y_k)^T y_k}, \tag{12}$$

the memory-less SR1 method with inverse approximation to the Hessian is obtained. From (12), after some simple algebraic manipulations, the memory-less SR1 search direction $d_{k+1} = -H_{k+1} g_{k+1}$ is obtained as

$$d_{k+1} = -g_{k+1} - \frac{(s_k - y_k)^T g_{k+1}}{(s_k - y_k)^T y_k}(s_k - y_k). \tag{13}$$

The main advantage of the memory-less SR1 update (13) is that for its implementation in computer programs only two scalar products $(s_k - y_k)^T g_{k+1}$ and $(s_k - y_k)^T y_k$ must be computed. This is very advantageous for solving large-scale problems. Observe that in the memory-less SR1 update the information on the Hessian approximation from the previous iteration is not accumulated to the current iteration.

Proposition 1. *Suppose that* $(s_k - y_k)^T y_k > 0,$ *then the memory-less SR1 search direction (13) is a descent direction.*

Proof. From (13) we get

$$g_{k+1}^T d_{k+1} = -\|g_{k+1}\|^2 - \frac{((s_k - y_k)^T g_{k+1})^2}{(s_k - y_k)^T y_k} < 0,$$

i.e. d_{k+1} given by (13) is a descent direction. ♦

Now, it can be noticed that the memory-less SR1 search direction (13) has three terms. The first one is the negative gradient $-g_{k+1}$, the last two terms involve s_k and y_k, both of them being multiplied by the same scalar. It is very simple to see that the search direction (13) satisfies the conjugacy condition, i.e. $y_k^T d_{k+1} = -s_k^T g_{k+1}$. Therefore, we can say that the memory-less SR1 method is a conjugate gradient method which satisfies the Dai and Liao [2001] conjugacy condition.

4. Memory-less BFGS Method

As we know the most effective quasi-Newton updating of the approximations to the Hessian is considered to be the BFGS formula [Nocedal, 1992], [Nocedal & Wright, 2006] where

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \tag{14}$$

which is a *rank-two update* that satisfies the secant equation (6). If H_k is the inverse approximation to the Hessian at iteration k, then by applying the Sherman-Morrison-Woodbury formula twice the following update to the inverse Hessian for BFGS is obtained

$$H_{k+1} = H_k - \frac{s_k y_k^T H_k + H_k y_k s_k^T}{y_k^T s_k} + \left(1 + \frac{y_k^T H_k y_k}{y_k^T s_k}\right)\frac{s_k s_k^T}{y_k^T s_k}. \tag{15}$$

The most important properties of BFGS are as follows. If H_k is positive definite, then also H_{k+1} given by (15) is positive definite for any k, provided that $y_k^T s_k > 0$ (which always is satisfied when the Wolfe line search (5) are satisfied). Therefore, if H_0 is chosen to be positive definite, then the rest of all the approximations H_k will also be positive definite. Also BFGS has the self-correcting property, i.e. if H_k incorrectly approximates the curvature of the minimizing function and this estimate slows down the iteration, then the inverse Hessian approximation will tend to correct itself in the next few iterations. The self-correcting property depends on the quality of the implementation of the Wolfe line search. For the Wolfe line search, always the initial value $\alpha = 1$ is tried and this produce superlinear convergence of the method. All these properties of BFGS update make this quasi-Newton method one of the best in this class.

Now considering $H_k = I$ in (15), the *memory-less BFGS method* is obtained

$$H_{k+1} = I - \frac{s_k y_k^T + y_k s_k^T}{y_k^T s_k} + \left(1 + \frac{y_k^T y_k}{y_k^T s_k}\right)\frac{s_k s_k^T}{y_k^T s_k}. \tag{16}$$

The corresponding search direction is $d_{k+1} = -H_{k+1} g_{k+1}$, where H_{k+1} is given by (16), i.e.

$$d_{k+1} = -g_{k+1} + \frac{(y_k^T g_{k+1})s_k + (s_k^T g_{k+1})y_k}{y_k^T s_k} - \left(1 + \frac{y_k^T y_k}{y_k^T s_k}\right)\frac{(s_k^T g_{k+1})s_k}{y_k^T s_k}. \tag{17}$$

Observe that the numerical computation of d_{k+1} from (17) involves only four scalar products: $y_k^T s_k$, $y_k^T y_k$, $y_k^T g_{k+1}$ and $s_{k+1}^T g_{k+1}$. Therefore, it very suitable for solving large-scale problems. It is worth seeing that the search direction corresponding to the memory-less BFGS updating has three terms. Besides, it is easy to prove that this search direction satisfies the Dai-Liao conjugacy condition, i.e. $y_k^T d_{k+1} = -s_k^T g_{k+1}$. We showed that also the memory-less SR1 method has this property of satisfying the conjugacy condition. Therefore, there is close connection between the quasi-Newton and the conjugate gradient methods. Shanno [1978a, 1978b] was the first who observed that *the conjugate gradient methods are precisely the quasi-Newton methods where the approximation to the inverse to the Hessian is restarted as the identity matrix at every iteration*.

There is a great difference between the memory-less SR1 method and the memory-less BFGS method. If in these methods the stepsize is computed by the Wolfe line search (5), then the memory-less BFGS method is well defined since at every iteration $y_k^T s_k \neq 0$. On the other hand, in case of SR1 method the Wolfe line search do not guarantee that $(s_k - y_k)^T y_k \neq 0$. However, both these methods are implemented in such a way that if the corresponding search directions are not defined, i.e. $y_k^T s_k = 0$, or $(s_k - y_k)^T y_k = 0$, then the search direction considered at that iteration is the negative gradient. In the following let us present the algorithms corresponding to the memory-less SR1 and to the memory-less BFGS methods.

5 The Memory-less SR1 and BFGS Algorithms

The memory-less algorithms corresponding to SR1 and BFGS are very simple and implement the Wolfe line search conditions (5) for stepsize computation and the corresponding search direction (13) and (17), respectively. In our numerical experiments in these algorithms we implemented the acceleration scheme developed by Andrei [2006, 2020a].

MM-SR1 and MM-BFGS Algorithms

	MM-SR1	**MM-BFGS**
1	Consider an initial point x_0. Set $k = 0$. Select some values for the Wolfe line search conditions σ and ρ with $0 < \rho < \sigma < 1$. Compute $g_0 = \nabla f(x_0)$ and set $d_0 = -g_0$. Select a sufficiently small parameters $\varepsilon > 0$ used in the criterion for stopping the iterations and $\varepsilon_A > 0$ used in acceleration scheme	
2.	Test a criterion for stopping the iterations: if $\|g_k\|_\infty \leq \varepsilon$ then stop the iterations, otherwise go to step 3	
3.	Compute the stepsize α_k using the standard Wolfe line search conditions	
4.	Update the variables $x_{k+1} = x_k + \alpha_k d_k$ and compute f_{k+1} and g_{k+1}. Compute	

	$s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$	
5.	Acceleration scheme: a) Compute: $z = x_k + \alpha_k d_k$, $g_z = \nabla f(z)$ and $y_k = g_k - g_z$ b) Compute: $\bar{a}_k = \alpha_k g_k^T d_k$, and $\bar{b}_k = -\alpha_k y_k^T d_k$ c) If $\lvert \bar{b}_k \rvert \geq \varepsilon_A$, then compute $\xi_k = -\bar{a}_k / \bar{b}_k$ and update the variables as $x_{k+1} = x_k + \xi_k \alpha_k d_k$. Compute f_{k+1} and g_{k+1}. Compute $y_k = g_{k+1} - g_k$ and $s_k = x_{k+1} - x_k$	
6.	Compute the search direction d_{k+1} as in (13). If the search direction (13) is not defined, then set $d_{k+1} = -g_{k+1}$	Compute the search direction d_{k+1} as in (17). If the search direction (17) is not defined, then set $d_{k+1} = -g_{k+1}$
7.	Restart iterations. If $g_{k+1}^T d_{k+1} > -10^{-3} \lVert g_{k+1} \rVert \lVert d_{k+1} \rVert$, then set $d_{k+1} = -g_{k+1}$	
8.	Consider $k = k+1$ and go to step 2 ♦	

Observe that the algorithm is equipped with an acceleration scheme (see step 5) introduced by Andrei [2006, 2009]. This scheme modifies the stepsize determined by the Wolfe line search conditions (5) in such a way as to improve the reduction of the minimizing function values along the iterations. It is proved that this acceleration scheme is linear convergent, but the reduction in the function value is significantly improved.
If f is bounded along the direction d_k, then there exists a stepsize α_k satisfying the Wolfe line search conditions (5). The first trial of the stepsize crucially affects the practical behavior of the algorithm. At every iteration $k \geq 1$, the starting guess for the step α_k in the line search is computed as $\alpha_{k-1} \lVert d_{k-1} \rVert / \lVert d_k \rVert$. Observe that in step 6 of the algorithms if the search direction for the memory-less SR1 method is not defined, i.e. if $\lvert (s_k - y_k)^T y_k \rvert$ is close to zero, or if the search direction for the memory-less BFGS is not defined, i.e. if $y_k^T s_k$ is close to zero, then the search direction is commuted to be the steepest descent. In our algorithm we introduced a restarting condition. If this restarting condition is satisfied, then the algorithm is restarted with the negative gradient. Some other restarting procedures may be implemented, but we are interested in seeing the performances of these algorithms implementing this restart condition.

6 Convergence of the memory-less SR1 method

In this section the global convergence of the algorithm is established under the following assumptions:

(A1) The level set $\Omega = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ is bounded, i.e. there exists a constant $B > 0$ such that for any $x \in \Omega$, $\lVert x \rVert \leq B$.

(A2) The function $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and its gradient is Lipschitz continuous in a neighborhood N of Ω, i.e. there exists a constant $L > 0$ such that $\lVert \nabla f(x) - \nabla f(y) \rVert \leq L \lVert x - y \rVert$, for any $x, y \in N$.

It is easy to see that under these assumptions, there exists a constant $\Gamma > 0$ such that $\|\nabla f(x)\| \leq \Gamma,$ for any $x \in \Omega.$

Although the search directions d_{k+1} generated by the memory-less algorithms SR1 and BFGS are always descent directions, in order to get the convergence of the algorithm we need to derive a lower bound for the stepsize α_k.

Proposition 2 *Suppose that d_k is a descent direction and ∇f satisfies the Lipschitz condition*

$$\|\nabla f(x) - \nabla f(x_k)\| \leq L\|x - x_k\|$$

for all x on the line segment connecting x_k and x_{k+1}, where L is a constant. If the line search satisfies the standard Wolfe conditions (5)*, then*

$$\alpha_k \geq \frac{1-\sigma}{L}\frac{|g_k^T d_k|}{\|d_k\|^2}. \tag{18}$$

Proof Subtracting $g_k^T d_k$ from both sides of (5.b) and using the Lipschitz condition, it follows that

$$(\sigma - 1)g_k^T d_k \leq (g_{k+1} - g_k)^T d_k \leq \alpha_k L \|d_k\|^2.$$

But d_k is a descent direction and $\sigma < 1,$ therefore (18) follows from the above inequality. ♦

Proposition 3 *Suppose that $(s_k - y_k)^T y_k > 0$ and $(s_k - y_k)(s_k - y_k)^T$ is a matrix bounded above in norm. Then, the memory-less SR1 update matrix H_{k+1} given by* (12) *is bounded above in norm.*

Proof. From (12), since the denominator $(s_k - y_k)^T y_k$ is strictly positive, the numerator $(s_k - y_k)(s_k - y_k)^T$ is bounded above in norm and $\|s_k\| = \|x_{k+1} - x_k\| \leq \|x_{k+1}\| + \|x_k\| \leq 2B,$ (see assumption (A1)), it follows that H_{k+1} given by (12) is bounded above in norm. ♦

Proposition 4 *Suppose that $(s_k - y_k)^T y_k > 0.$ Then the memory-less SR1 search direction* (13)*, is a descent direction, that is $g_{k+1}^T d_{k+1} < 0.$ Besides, $\|d_{k+1}\| \leq D,$ for some $D > 0.$*

Proof. Observe that the numerator of the update (13) is a rank-one positive semidefinite matrix and the denominator of (13) is strictly positive. Therefore the update matrix H_{k+1} given by (13) is positive semidefinite. Without loss of generality, H_k is positive definite, hence, H_{k+1} given by (13) is positive definite. Thus, $g_{k+1}^T d_{k+1} = -g_{k+1}^T H_{k+1} g_{k+1} < 0.$

The boundedness of d_{k+1} may be established as follows. If the restart condition is satisfied, then $d_{k+1} = -g_{k+1}.$ Therefore, $\|d_{k+1}\| = \|g_{k+1}\| < D,$ where $D = \Gamma$ by assumption (A2). On the other hand, for non-restart iterations the search direction is computed as $d_{k+1} = -H_{k+1} g_{k+1},$ where H_{k+1} is given by (12). In this case, by Proposition 3 the sequence of the Hessian matrices given by (12) remains bounded above in norm. Therefore, $\|d_{k+1}\| = \|-H_{k+1}g_{k+1}\| \le \|H_{k+1}\|\|g_{k+1}\|$, which is bounded above by some constant $D > 0$. ♦

Theorem 1 *Suppose that the assumptions (A1) and (A2) are satisfied. If* f *is Lipschitz continuous, then* $\lim_{k\to\infty}\|g_k\| = 0$.

Proof The assumptions and the above Propositions 2, 3 and 4 show that the algorithm MM-SR1 is globally convergent to a point in which the first-order optimality conditions are satisfied. ♦

7 Numerical results

In this section we report some numerical results obtained with memory-less SR1 and with memory-less BFGS methods for solving unconstrained optimization problems. The memory-less SR1 method is given by (4), where the stepsize is computed by the Wolfe line search (5) and the search direction is given by (13). The memory-less BFGS method is given by (4), where the stepsize is computed as in (5) and the search direction is given by (17).

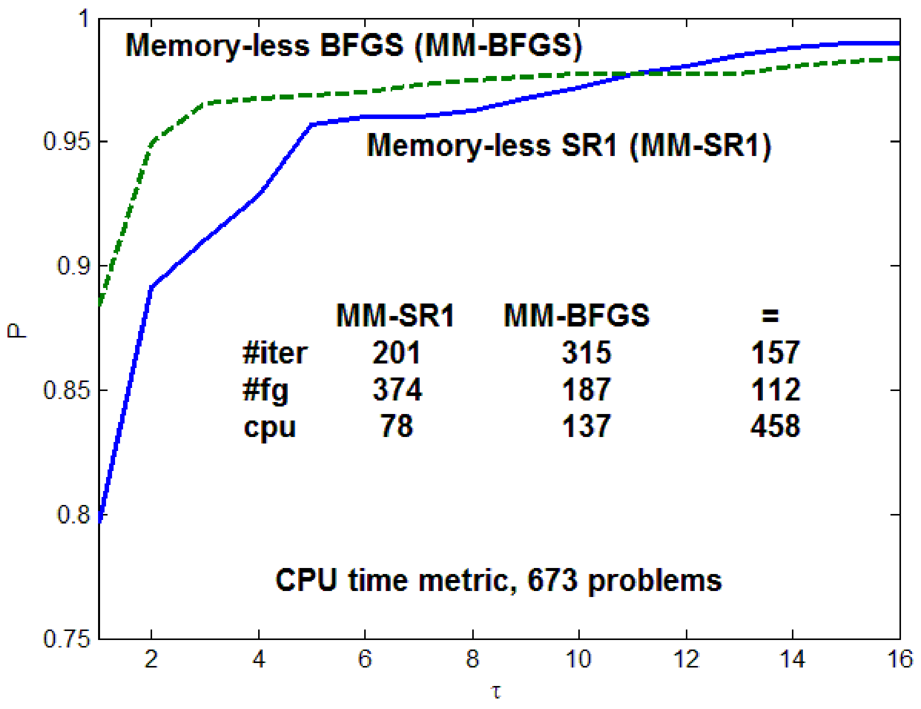
The codes are written in Fortran and compiled with f77 (default compiler settings) on a Workstation Intel Pentium 4 with 1.8 GHz. We selected a number of 80 large-scale unconstrained optimization test functions in generalized or extended form we presented in [Andrei, 2020a]. The vast majority of these problems are taken from CUTE collection [Bongartz, Conn, Gould and Toint, 1995]. For each test function we have taken ten numerical experiments with an increasing number of variables. Therefore, a number of 800 unconstrained optimization problems have been considered. The algorithms used in all numerical test implement the Wolfe line search conditions (5) with $\rho = 0.0001$, $\sigma = 0.8$ and the same stopping criterion $\|g_k\|_\infty \le 10^{-6}$, where $\|.\|_\infty$ is the maximum absolute component of a vector. In all algorithms, we considered in our numerical studies, the maximum number of iterations is limited to 10000, while the maximum number of function and its gradient evaluations is limited to 10000.

The comparisons of algorithms are given in the following context. Let f_i^{ALG1} and f_i^{ALG2} be the optimal value found by ALG1 and ALG2, for problem $i = 1,\ldots,800$, respectively. We say that, in the particular problem i, the performance of ALG1 was better than the performance of ALG2 if:

$$\left| f_i^{ALG1} - f_i^{ALG2} \right| < 10^{-3} \tag{19}$$

and the number of iterations (#iter), or the number of function-gradient evaluations (#fg), or the CPU time of ALG1 was less than the number of iterations, or the number of function-gradient evaluations, or the CPU time corresponding to ALG2, respectively.

In the first set of numerical experiments let us compare MM-SR1 versus MM-BFGS for solving this set of unconstrained minimization problems with the number of variables in the range [100, 1000]. Figure 1 shows the Dolan and Moré [2002] performance profiles of these algorithms.



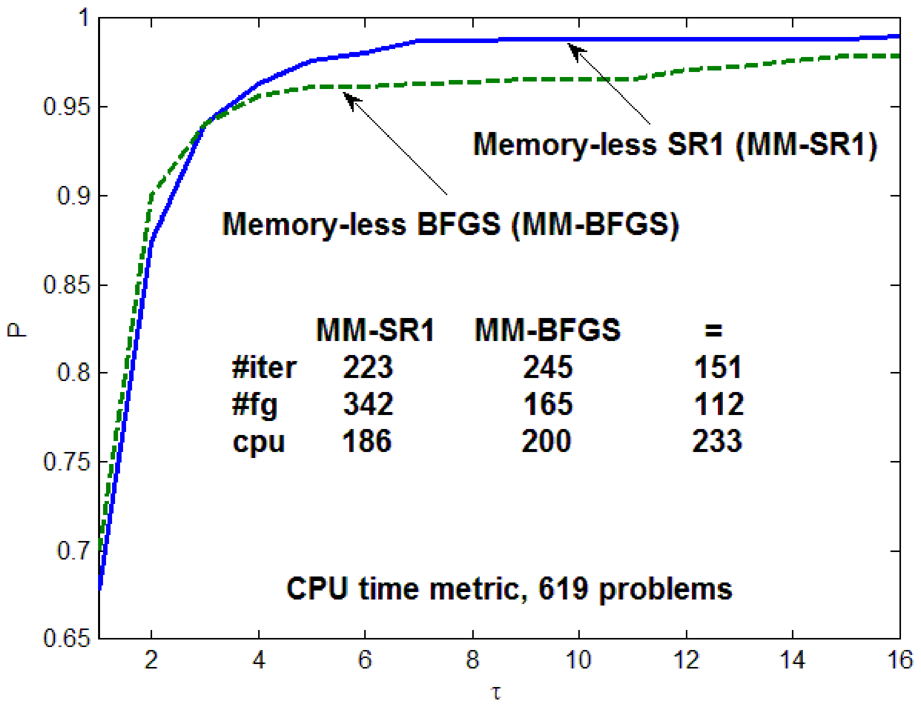
	MM-SR1	MM-BFGS	=
#iter	201	315	157
#fg	374	187	112
cpu	78	137	458

Fig. 1. Memory-less SR1 versus memory-less BFGS, range [100, 1000]

In a performance profile plot, the top curve corresponds to the method that solved the most problems in a time that was within a given factor of the best time. The percentage of the test problems for which a method is the fastest is given on the left axis of the plot. The right side of the plot gives the percentage of the test problems that were successfully solved by these algorithms, respectively. Mainly the left side of the plot is a measure of the efficiency of an algorithm, while the right side is a measure of the robustness of an algorithm.

From Figure 1 we see that memory-less BFGS algorithm is more efficient than the memory-less SR1 method, but MM-SR1 is slightly more robust. Indeed, comparing MM-SR1 versus MM-BFGS subject to the number of iterations, we see that MM-BFGS was better in 315 problems (i.e. it achieved the minimum number of iterations in 315 problems). MM-SR1 was better in 201 problems and they achieved the same number of iterations in solving 157 problems, etc. Out of 800 problems considered in this numerical experiment, only for 673 problems does the criterion (19) hold. Subject to the CPU computing time, from Figure 1 we see that MM-BFGS was faster in 137 problems, while MM-SR1 was faster only in 78 problems. Notice that both MM-SR1 and MM-BFGS use the same implementation of the standard Wolfe line search (5) based on cubic interpolation [Shanno, 1983], as well as the same optimization conditions.

In the second set of numerical experiments, Figure 2 presents the performance profiles of MM-SR1 versus MM-BFGS for solving unconstrained optimization problems from our collection, where this time the number of variables is in the range [1000, 10000].



	MM-SR1	MM-BFGS	=
#iter	223	245	151
#fg	342	165	112
cpu	186	200	233

Fig. 2. Memory-less SR1 versus memory-less BFGS, range [1000, 10000]

Observe that MM-SR1 is more robust than MM-BFGS. Subject to the CPU time metric, MM-SR1 was faster in 186 problems, but MM-BFGS was faster in 200 problems. Practically, these memory-less quasi-Newton algorithms have the same efficiency.

In the third set of numerical experiments let us compare the performances of MM-SR1 versus BFGS from CONMIN [Shanno and Phua, 1976]. CONMIN package includes two optimization algorithms: a BFGS preconditioned conjugate gradient one and a variant of the BFGS quasi-Newton algorithm. Figure 3 shows the performance profiles of these algorithms for solving 800 unconstrained optimization problems from our collection with the number of variables in the range [100, 1000].

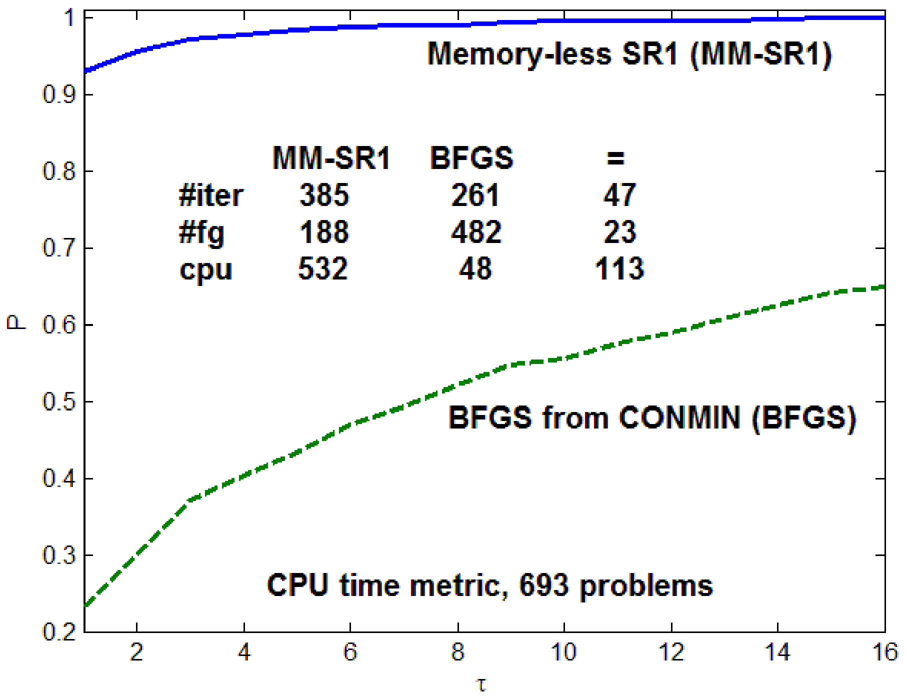


Fig. 3. Memory-less SR1 versus BFGS from CONMIN, range [100, 1000]

Obviously, MM-SR1 is more efficient and more robust than BFGS from CONMIN, one of the best implementation of this quasi-Newton method. Observe that subject to the CPU computing time MM-SR1 was faster in 532 problems, while BFGS from CONMIN was faster only in 48 problems. The BFGS from CONMIN is a variable metric method with initial scaling which approximately needs $n^2/2+11n/2$ double precision words of working storage. In comparison MM-SR1 requires approximately $6n$ double precision words of working storage. BFGS requires more memory and involves a greater computational effort. It is worth showing the performances of MM-SR1 and of BFGS from CONMIN for solving the problem Freudenstein & Roth with $n=1000$ variables. BFGS from CONMIN gives a solution for which $f(x^*)=24492.126839,$ in 20 iterations, 21 evaluations of the function and its gradient and 0.44 seconds. On the other hand, MM-SR1 gives a solution with the same value of the minimizing function, but in 6 iterations, 19 evaluations of the function and its gradient and 0 seconds. We emphasize that at every iteration BFGS from CONMIN update an approximation to the Hessian by accumulating the information from the previous iterations. On the other hand memory-less MM-SR1 algorithm at every iteration update the identity matrix (see (12)). Obviously, MM-SR1 doesn't accumulate the information from iteration to iteration when updating the approximation to the Hessian. However, MM-SR1 is way more efficient and more robust than BFGS from CONMIN.

In the last set of numerical experiments let us present comparisons between MM-SR1 and MM-BFGS algorithms for solving five applications from the MINPACK-2 test problem collection [Averick, Carter, Moré, & Xue, 1992]. MINPACK-2 contains applications from different fields, such as: elasticity, fluid dynamics, combustion, lubrication, molecular conformation, nondestructive testing, chemical kinetics, etc. Table 1, presents the applications considered in our numerical experiments, as well as the values of their parameters. The infinite-dimensional version of these problems is transformed into a finite element approximation by triangulation. Thus a finite-dimensional minimization problem is obtained whose variables are the values of the piecewise linear function at the vertices of the triangulation. The discretization steps are $nx = 200$ and $ny = 200$, thus obtaining minimization problems with $nx \times ny = 40,000$ variables.

Table 1
Applications from the MINPACK-2 collection

A1	Elastic–plastic torsion [Glowinski, 1984, pp. 41–55], c = 5
A2	Pressure distribution in a journal bearing [Cimatti, 1977], b = 10, ε = 0.1
A3	Optimal design with composite materials [Goodman, Kohn, & Reyna, 1986], λ = 0.008
A4	Steady-state combustion [Aris, 1975, pp. 292–299], [Bebernes, & Eberly, 1989], λ = 5
A5	Minimal surfaces with Enneper conditions [Nitsche, 1989, pp. 80–85]

The performances of the accelerated memory-less SR1 method (MM-SR1) versus the accelerated memory-less BFGS (MM-BFGS) method are given in Table 2, where #iter is the number of iterations, #fg is the number of function and its gradient evaluations, #ig is the number of iterations in which the search direction is the negative gradient and cpu is the CPU time computing for solving these applications.

Table 2
Performances of MM-SR1 versus MM-BFGS (40,000 variables, CPU seconds)

	MM-SR1				**MM-BFGS**			
	#iter	#fg	cpu	#ig	#iter	#fg	cpu	#ig
A1	13138	26297	326.04	1	6711	20133	180.56	6611
A2	66930	133892	1343.10	0	9312	27937	298.50	8883
A3	88142	176330	4915.73	822	861	2584	53.28	15
A4	49631	99287	3708.33	1	666	1997	140.00	416
A5	11370	22775	320.38	1	2177	6520	148.46	199
TOTAL	**229211**	**458481**	**10613.58**	**825**	**19727**	**59171**	**820.80**	**16124**

From Table 2 we see that MM-BFGS is more efficient than MM-SR1. For example, subject to the CPU time metric, for solving all these five applications, MM-BFGS needs 820.80 seconds, while MM-SR1 needs 10613.50 seconds, i.e. MM-BFGS is 12.93 times faster than MM-SR1. It is worth mentioning that for solving all these five applications, in MM-SR1 out of 229211 iterations, the negative gradient was used only for 825 iterations (i.e. 0.36%), while in MM-BFGS out 19727 iterations, the negative gradient was used for 16124 iterations (i.e. 81.73%). Even that, for solving all these five applications, in the vast majority of iterations MM-BFGS used the negative gradient as the search direction, the performances of MM-BFGS are better. This is because the MM-BFGS retains the self-correcting property of BFGS, i.e. if H_k incorrectly approximates the curvature of the minimizing function, mainly due to the line search, then the inverse Hessian approximation will tend to correct itself in the next iterations. MM-SR1 does not have the self-correcting property and this is the reason why its performances are modest.

8 Conclusions

This paper presents two quasi-Newton methods, SR1 and BFGS, with memory less. At every iteration, the memory-less SR1 and the memory-less BFGS initialize the approximation to the Hessian with the identity matrix. Therefore, at the current iteration these methods do not accumulate the information about the Hessian from the previous iteration. The memory-less technique is simple and efficient for adapting the quasi-Newton methods for solving large-scale problems. Our numerical experiments, involving 800 large-scale unconstrained optimization problems with different structures and complexities, show that the memory-less SR1 method is more robust than the limited-memory BFGS method, and more efficient and more robust than the BFGS from CONMON package. It seems that the accuracy of the approximation to the Hessian is not a crucial ingredient in efficiency and robustness of quasi-Newton methods.

References

Andrei, N., (2006). An acceleration of gradient descent algorithm with backtracking for unconstrained optimization. *Numerical Algorithms, 42*(1), 63-73.

Andrei, N., (2009). Acceleration of conjugate gradient algorithms for unconstrained optimization. *Applied Mathematics and Computation, 213*(2), 361-369.

Andrei, N., (2020a). *Nonlinear conjugate gradient methods for unconstrained optimization.* Springer Optimization and Its Applications 158.

Aris, R., (1975). The Mathematical Theory of Diffusion and Reaction in Permeable Catalysts, Oxford.

Averick, B.M., Carter, R.G., Moré, J.J. & Xue, G.L., (1992). *The MINPACK-2 test problem collection,* Mathematics and Computer Science Division, Argonne National Laboratory, Preprint MCS-P153-0692.

Bebernes, J., & Eberly, D., (1989). Mahematical Problems from Combustion Theory, in: Applied Mathematical Sciences, vol. 83, Springer-Verlag.

Benson, H.Y., & Shanno, D.F., (2018). Cubic regularization in symmetric rank-1 quasi-Newton methods. *Math. Progr. Comput. 10,* 457-486.

Bongartz, I., Conn, A.R., Gould, N.I.M., Toint, Ph.L., (1995). CUTE: constrained and unconstrained testing environments. *ACM Transactions on Mathematical Software, 21*, 123-160.

Broyden, C.G., (1967) Quasi-newton methods and their applications to function minimization. *Mathematics of Computation 21*(99), 368-381.

Broyden, C.G., (1970). The convergence of a class of double-rank minimization algorithms. I. General considerations. *Journal of the Institute of Mathematics and Its Applications, 6,* 76-90.

Chen, H., Lam, W.H., & Chan, S.C., (2019). On the convergence analysis of cubic regularized symmetric rank-1 quasi-Newton method and the incremental version in the application of large-scale problems. *IEEE Access,* volume 7, 114042-114059.

Cimatti, G., (1977). On a problem of the theory of lubrication governed by a variational inequality, *Applied Mathematics and Optimization 3*, 227–242.

Conn, A.R., Gould, N.I.M., & Toint, Ph.L., (1991). Convergence of quasi-newton matrices generated by the symmetric rank one update. *Mathematical Programming 50*(1-3), 177-195.

Davidon, W.C., (1959). *Variable metric method for minimization.* (Research and Development Report ANL-5990. Argonne National Laboratories.)

Davidon, W.C., (1991). Variable metric method for minimization. *SIAM J. Optim. 1*(1), 1-17.

Dolan, E.D., & Moré, J.J., (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming, 91*, 201-213.

Fiacco, A.V., & McCormick, G.P., (1968). Nonlinear programming: sequential unconstrained minimization techniques. Research Analysis Corporation, McLean Virginia. Republished in 1990 by SIAM, Philadelphia.

Fletcher, R., & Powell, M.J.D., (1963). A rapidly convergent descent method for minimization, *Computer Journal*, 163–168.

Fletcher, R., (1970). A new approach to variable metric algorithms. *The Computer Journal, 13*, 317-322.

Glowinski, R., (1984). *Numerical Methods for Nonlinear Variational Problems*, Springer-Verlag, Berlin.

Goldfarb, D., (1970). A family of variable metric method derived by variation mean. *Mathematics of Computation, 23*, 23-26.

Goodman, J., Kohn, R., & Reyna, L., (1986). Numerical study of a relaxed variational problem from optimal design, *Computer Methods in Applied Mechanics and Engineering 57*, 107–127.

Huang, H.Y., (1970). Unified approach to quadratically convergent algorithms for functions minimization. *Journal of Optimization Theory and Applications 5*(6), 405-423.

Kelley, C.T., & Sachs, E.W., (1998). Local convergence of the symmetric rank one iteration. *Computational Optimization and Applications, 9*, 43–63.

Khalfan, H.F., Byrd, R.H., & Schnabel, R.B., (1993). A theoretical and experimental study of the symmetric rank-one update. *SIAM J. Optimization 3*(1), 1-24.

Nitsche, J.C.C., (1989). *Lectures on Minimal Surfaces*, Vol. 1, Cambridge University Press.

Nocedal, J., (1992). Theory of algorithms for unconstrained optimization. *Acta Numerica, 1*, 199-242.

Nocedal, J., & Wright, S.J., (2006). *Numerical optimization.* Springer Series in Operations Research. Springer Science+Business Media, New York, Second edition, 2006.

Powell, M.J.D., (1970). A new algorithm for unconstrained optimization. In: J.B. Rosen, O.L. Mangasarian & K. Ritter (Eds.) *Nonlinear Programming*. Academic Press, New York, 31-66.

Shanno, D.F., (1970). Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation, 24*, 647-656.

Shanno, D.F., (1978a). Conjugate gradient methods with inexact searches, *Mathematics of Operations Research, 3*, 244-256.

Shanno, D.F., (1978b). On the convergence of a new conjugate gradient algorithm, *SIAM Journal on Numerical Analysis*, *15*, 1247-1257.

Shanno, D.F., (1983). *CONMIN – A Fortran subroutine for minimizing an unconstrained nonlinear scalar valued function of a vector variable* x *either by the BFGS variable metric algorithm or by a Beale restarted conjugate gradient algorithm*. Private communication, October 17, 1983.

Shanno, D.F., & Phua, K.H., (1976). Algorithm 500. Minimization of unconstrained multivariable functions. *ACM Transactions on Mathematical Software*, *2*, 87-94.

Wolfe, P., (1969). Convergence conditions for ascent methods. *SIAM Review*, *11*, 226-235.

Wolfe, P., (1971). Convergence conditions for ascent methods. II: Some corrections. *SIAM Review*, *13*, 185-188.

-----ooooOooooo-----