

Minimization Algorithms Based on Supervisor and Searcher Cooperation^{1,2,3}

W. LIU⁴ AND Y. H. DAI⁵

Communicated by P. M. Pardalos

Abstract. In the present work, we explore a general framework for the design of new minimization algorithms with desirable characteristics, namely, supervisor-searcher cooperation. We propose a class of algorithms within this framework and examine a gradient algorithm in the class. Global convergence is established for the deterministic case in the absence of noise and the convergence rate is studied. Both theoretical analysis and numerical tests show that the algorithm is efficient for the deterministic case. Furthermore, the fact that there is no line search procedure incorporated in the algorithm seems to strengthen its robustness so that it tackles effectively test problems with stronger stochastic noises. The numerical results for both deterministic and stochastic test problems illustrate the appealing attributes of the algorithm.

Key Words. Robust algorithms, noisy optimization, gradient algorithms, stochastic approximations.

1. Introduction

In practical applications, various types of noises are present in most available data. Here, noises are understood in a broad sense. In some cases,

¹A part of the work was reported at the 19th IFIP TC7 Conference on System Modelling and Optimization, Cambridge, England, 1999.

²The paper was initiated while the second author was visiting Canterbury Business School, University of Kent. He thanks the faculty of Social Sciences of the University for support and hospitality.

³The authors express their sincere thanks to Mr. K. Sirlantzis for helpful comments and suggestions, important contributions to the SSC-SABB results in the deterministic case, and excellent work on the stochastic case.

⁴Canterbury Business School, University of Kent, Canterbury, England.

⁵State Key Laboratory of Scientific and Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences, Beijing, PRC.

noises can be ignored, but more often than not, they play an important role in the mathematical modeling of applications. The purpose of this work is to develop faster and reliable algorithms for unconstrained optimization problems with stronger noise,

$$\min f(x), \quad \text{where } f(x) = F(x) + \epsilon,$$

where F is the underlying exact mathematical model and either (i) or (ii) below characterizes the noise ϵ :

- (i) ϵ is stochastic noise, which may depend on x . In this case a desirable algorithm should be faster when $\epsilon = 0$, and still able to solve the problem with larger noises.
- (ii) ϵ is some kinds of deterministic residual, e.g., from inexact computing. In such case, it is possible that ϵ is negligible in function value evaluation, but significant in gradient estimation. Again, a desirable algorithm should be efficient when noises are smaller and still able to solve the larger noise case.

We anticipate that a good algorithm has to be efficient when noise is negligible and reliable when it is not, since it is difficult to control the exact level of noise in applications.

In this paper, whenever we refer to the function value or gradient of f at a point $x \in R^n$, we mean some estimation of the value of F or gradient of F at the point x . When we refer to a minimizer of f , we really mean a minimizer of F . When $\epsilon = 0$, the estimators are assumed equal to the corresponding exact values.

Before introducing our algorithms, we would like to explain the motivation behind them. Let us first assume $\epsilon = 0$ and have a closer look at a classical minimization algorithm: Given x_0, t_0, v_0 , compute

$$x_{k+1} = x_k - t_k v_k, \quad k = 0, 1, 2, \dots, \quad (1)$$

where t_k and v_k are the steplength and search direction. In many cases, the following algorithm (search engine) is locally convergent:

$$x_{k+1} = x_k - v_k.$$

To a large extent, it actually decides the speed of the algorithm (1). The global convergence of the algorithm (1) is normally ensured by selecting the steplengths $\{t_k\}$ via a line search procedure, for instance: (i) exact monotone type; (ii) inexact monotone type, e.g., Armijo–Goldstein and Wolfe–Powell (Refs. 1–2); (iii) inexact nonmonotone type, e.g., Grippo–Lampariello–Lucidi (Ref. 3).

Most existing line search procedures are not found to be very robust when applied to stochastic optimization, where strong noises may be present

in the evaluation of the objective functions, or to nonsmooth optimization, where the objective functions are in general only Lipschitz. For instance, in stochastic optimization, an exact line search could prove to be very expensive and unstable due to noises. Most existing inexact line search methods, though may be less expensive, seem to have similar problems. For example, in stochastic optimization, the estimated gradient contains normally much stronger noises than those present in the measurement of an objective function. Hence, the line search results may not be consistent. As a matter of fact, the existing line search procedures fail frequently, as it will be seen in our numerical experiments. For nonsmooth objective functions, line search methods are not applicable in general.

There has been extensive research in developing efficient algorithms for the above noisy optimization problems. An obvious approach is to use

$$\hat{f}(x_k) = \sum_{i=1}^n f(x_k)/n$$

to evaluate the objective function value, at least for the stochastic case. If the noises have zero mean, this surely can increase the accuracy of estimation of the function value. Then, existing deterministic optimization algorithms may be applicable. However, this procedure is in general expensive. It will not work for nonstochastic noises.

A very popular algorithm in engineering computation is the stochastic approximation (SA) algorithm, which has been used widely in various applications with stronger (stochastic) noises since the 1950s. It starts from two cornerstone papers (Refs. 4–5) and has received extensive references in the literature; see for instance, Refs. 6–9 and the references cited therein. In this algorithm, there is no line search at all. It uses a prefixed stepsize $\{t_k\}$, e.g.,

$$t_k = 1/k, \quad t_k = 1/k^{0.5}, \quad \text{or } t_k = 0.001.$$

SA proves to be very robust, and its global convergence has been established under various assumptions on the noises. The most common criticisms are that SA is in general very slow and that it is difficult to select suitable $\{t_k\}$, as it will be shown later on in some examples. For deterministic problems without noises, SA has been known to be very slow in comparison with efficient gradient algorithms like the conjugate gradient (CG) method.

There have been many improvements on SA since the 1950s. For example, a few adaptive or second-order SA methods were proposed (see Refs. 6–9), though most seem either to be expensive or to bring only marginal improvements. There are also quasi-Newton type and trust-region type algorithms (see the Uryas'ev work in Ref. 10 and trust-region methods in Ref. 11), though the algorithms are in general quite expensive for larger problems.

There are also attempts to generalize the standard line search procedures to the stochastic case. For instance, in Ref. 12, an Armijo-type line search with restarts is proposed. The basic idea is that the line search is allowed to restart if it fails. Since the estimated objective function values are in general different each time they are resampled, this line search should eventually go through if the estimation of the gradient is quite accurate. This idea has not yet been used widely in applications. In our experiments, it was found that it may not work efficiently.

To illustrate our point, we present some numerical results for a very simple stochastic quadratic minimization problem with

$$F(x) = \sum_{i=1}^n ix_i^2 + \sum_{i=1}^{n-1} x_i x_{i+1}, \quad n = 50, \quad \epsilon = 0.1N(0, 1),$$

where $N(0, 1)$ is the standard normal random variable. It is found that SA with $t_k = 0.001$ takes on average 128 sec of CPU time to solve it and SA with $t_k = 0.1/k$ simply fails. It is found that the Grippo–Lampariello–Lucidi line search (with 100 restarts) fails also. The stopping rule and more details will be explained later on.

There are other methods which do not use derivatives of the objective functions, e.g., the well-known response surface method (RSM); see Ref. 13 for some new developments and also see Ref. 14 for a survey of derivative-free algorithms in the deterministic case. There is vast literature in this area; for example, see Refs. 15–16 for stochastic problems. However, in general, these algorithms are not as efficient as those using derivative information, when available. Some techniques for optimizing stochastic discret-event systems via simulation are reviewed in Ref. 11.

In this and forthcoming papers, we propose a class of new algorithms whose general principle will be discussed in the next section. Line search procedures can be eliminated completely from these algorithms, and yet we do not use prefixed steps all the time. It seems that this class of algorithms can be both efficient and robust. In this paper, we examine in detail only one algorithm from the class. Both theoretical analysis and numerical experiments show that this algorithm is quite fast in the deterministic case with low noise, comparable with the conjugate gradient algorithm, for example. Furthermore, numerical tests show that the new algorithm is very robust and can tackle many noisy optimization problems where SA fails.

The plan of this paper is as follows. In Section 2, we discuss the supervisor and searcher cooperation framework within which our new algorithms are formulated. In Section 3, a particular algorithm is proposed via this framework. In Section 4, we establish some convergence theorems and analyze the speed of the algorithm for the deterministic case without noise. In

Section 5, numerical test results are presented for both deterministic and stochastic problems.

2. Supervisor-Searcher Cooperation

Let

$$f = F + \epsilon,$$

where F is a continuous function on R^n and is bounded below. We are interested in finding local minimizers of f , that is, F . For given x_0, x_1, \dots, x_m , assume that we have an iterative algorithm, called the search engine (SE):

$$x_{k+1} = x_k - \text{se}_k(x_k, x_{k-1}, \dots, x_{k-m}, k, f), \quad k = m, m+1, \dots$$

The particular form of se_k means that it depends on k and the values of $\{f(x_{k-i})\}_{i=0}^m$, $\{\nabla f(x_{k-i})\}_{i=0}^m$, $\{H(x_{k-i})\}_{i=0}^m$, etc, where $H(x)$ is the Hessian matrix of f at the point x .

Suppose that $\epsilon = 0$ and that this algorithm is convergent to a local minimizer of f provided the starting points are very close to the minimizer. To make the algorithm convergent globally, it is classic to introduce into it a line search procedure, monotone or nonmonotone, exact or inexact. However, as mentioned before, a line search procedure is in general sensitive to the smoothness of the function and the accuracy of evaluation of the function value. Therefore, the resulting algorithm, though convergent globally, may not be robust enough to deal with stochastic or nonsmooth optimization problems, which are becoming increasingly important in practical applications.

The essential idea adopted here is to employ an alternative globally convergent and robust iterative algorithm to supervise and therefore to safeguard the convergence of the SE algorithm. This supervising algorithm will be referred to as the supervisor (SR). Then, one may obtain a globally convergent and robust algorithm. Assume that the supervisor algorithm (SR) reads as follows: Given x_0, x_1, \dots, x_m , compute

$$x_{k+1} = x_k - \text{sr}_k(x_k, x_{k-1}, \dots, x_{k-m}, k, f), \quad k = m, m+1, \dots$$

In general, an SR algorithm is slower but robust and an SE algorithm is faster but only locally convergent. Therefore, they have to cooperate in order to work efficiently. We propose a supervision principle based on supervisor and searcher cooperation (SSC). According to this principle, the supervisor intervenes only when it believes that the performance of the search engine is not satisfactory, while the search engine undertakes most

of the (solution) searching work. For the resulting algorithm, to a large extent global convergence may be ensured by the supervisor, but the speed is decided by the search engine.

There are various ways to implement the cooperation principle. The following one is particularly simple. Assume $f \geq 0$. Given x_0, x_1, \dots, x_m , for $k = m, m+1, m+2, \dots$, define the following (SSC) algorithm:

$$x_{k+1} = \begin{cases} x_k - \text{sr}_k, & \text{if } T_k f(x_k - \text{sr}_k) \leq f(x_k - \text{se}_k), \\ x_k - \text{se}_k, & \text{otherwise,} \end{cases}$$

where $\{T_k\}$ is a given sequence of nonnegative real numbers.

This new algorithm may be very different from the parent algorithms, even when $\epsilon = 0$. The algorithm actually switches between the two original algorithms. For instance, assuming $\epsilon = 0$ and assuming to use the Newton search engine, then this algorithm takes only one step to find the minimizer for a convex quadratic objective function when taking $T_k \equiv 1$. This may be very different from the SR algorithm. On the other hand, it is not clear that the well known n -step convergence property might still hold for the SSC algorithm when using the conjugate gradient algorithm as search engine.

It is clear that the behaviors of an SSC algorithm depend not only on these of the SR and SE algorithms, and but also on the degree of the supervision, which is decided by the sequence $\{T_k\}$. For instance, if we take $T_k = 0$ or $T_k = \infty$, there will be no action of search engine or supervisor, assuming $f(x_k - r_k) > 0$. Also, by taking $\text{se}_k = 0$, the SSC algorithm becomes the SR algorithm. We have little interest in such degenerate situations. However, this does indicate the wide range of algorithms covered by the SSC algorithm. It seems clear that taking a larger T_k will force the SSC algorithm to use more SE iterations, and therefore may increase the overall speed of the SSC algorithm. However, if T_k is too large, the supervision may be too weak, and therefore the SSC algorithm may not be robust or even convergent globally. These issues will be examined more closely in the next section. It is also possible to let the algorithm check the switch every two or more iterations, to form a multistep SSC algorithm. Again, this could save much computational work, but weakens the cooperation.

Remark 2.1. Note that as far as minimization is concerned, one can always assume that $f \geq 0$ by adding a positive constant to the original function. Or one can use the following (SSC) algorithm for the general case:

$$x_{k+1} = \begin{cases} x_k - \text{sr}_k, & \text{if } T_k^{\text{sign}(f(x_k - \text{sr}_k))} f(x_k - \text{sr}_k) \leq f(x_k - \text{se}_k), \\ x_k - \text{se}_k, & \text{otherwise,} \end{cases}$$

where $\{T_k\}$ is a given sequence of nonnegative real numbers. Then, all the above observations apply to the general case.

There are many possible candidates for SR algorithms. In general, they are expected to be simple and robust with global convergence property. As for the SE algorithms, we may use various fast algorithms like the Newton algorithm, BFGS algorithm, and a fast gradient methods like the conjugate gradient method.

To motivate the investigation, we study only one pair of SR and SE algorithm in this paper, and leave more general cases to subsequent papers. In Section 3, we use the stochastic approximation algorithm as supervisor and a fast gradient method as searcher.

It is clear that there are many other ways to design minimization algorithms within the SSC framework. For instance, one can use the following rule:

$$x_{k+1} = \begin{cases} x_k - \text{sr}_k, & \text{if } T_k[f(x_k - \text{sr}_k) - f(x_k)] \leq f(x_k - \text{se}_k) - f(x_k), \\ x_k - \text{se}_k, & \text{otherwise,} \end{cases}$$

where $\{T_k\}$ is a given sequence of nonnegative real numbers. However, we shall not examine these implementations in this paper.

3. SSC Gradient Algorithm without Line Search

In this section, we propose a SSC gradient algorithm to compute a local minimizer of f . This algorithm uses a stochastic approximation (SA) algorithm as SR and the Barzilai–Borwein (BB) gradient algorithm as SE.

Let $\{t_k\}$, $k = 0, 1, 2, \dots$, be such that

- (i) $t_k > 0$,
- (ii) $\sum_{k=1}^{\infty} t_k = +\infty$.

These conditions are assumed throughout the paper. It should be noted that we do not assume that $t_k \rightarrow 0$ as $k \rightarrow \infty$.

In the following, we shall take

$$\text{sr}_k = t_k g_k, \quad \text{where } g_k = \nabla f(x_k),$$

that is, an estimate of $\nabla F(x_k)$. Therefore, the supervisor (SR) is the stochastic approximation algorithm: for given x_0 ,

$$x_{k+1} = x_k - t_k g_k, \quad k = 0, 1, 2, \dots$$

It is well known that the SA method is simple but very robust: it has been used widely for stochastic optimization problems. In general, it is too slow

to achieve higher computational accuracy, and it is difficult to select the sequence $\{t_k\}$ for a particular problem. The readers are referred to Refs. 6–9 for more details.

The searcher is based on the following BB gradient algorithm: Given x_0 , compute

$$x_{k+1} = x_k - \alpha_k g_k, \quad k = 0, 1, 2, \dots,$$

where $\alpha_0 = 1$ and, for $k \geq 1$,

$$\alpha_k = |x_k - x_{k-1}|^2 / (x_k - x_{k-1})' [\nabla f(x_k) - \nabla f(x_{k-1})].$$

The steplength α_k is referred to as the BB stepsize and was proposed first by Barzilai and Borwein in Ref. 18 for problem with $\epsilon = 0$. The BB algorithm is further studied in Ref. 19. In computation, $\{\alpha_k\}$ is normally forced to be bounded.

When $\epsilon = 0$, the BB algorithm has been shown to be locally convergent and R-superlinear for two-dimensional convex quadratic functions, and much faster than the steepest descent method. In Ref. 20, a nonmonotone line search is added to the BB algorithm in order to make it globally convergent. It is reported in Ref. 20 that, when $\epsilon = 0$, the resulting algorithm GBB algorithm is rather fast, comparable to the CG algorithm in many cases. In fact, it is faster than the CG algorithm for some large scale convex optimization problems. This motivates us to use the BB algorithm as the search engine.

We are now in the position to define the SSC gradient algorithm. Let $x_0 \in R^n$ and $\alpha_0 = 1$ be given. Let $T_k \geq 0$ be given for $k = 0, 1, 2, \dots$. Assume that $f \geq 0$. Then, define the following gradient optimization algorithm (SSC-SABB):

$$x_{k+1} = \begin{cases} x_k - t_k g_k, & \text{if } T_k f(x_k - t_k g_k) \leq f(x_k - \alpha_k g_k), k = 0, 1, 2, \dots, \\ x_k - \alpha_k g_k, & \text{otherwise.} \end{cases}$$

If f is not nonnegative, then the above definition may be modified as

$$x_{k+1} = \begin{cases} x_k - t_k g_k, & \text{if } T_k^{\text{sign}}(f(x_k - t_k g_k)) f(x_k - t_k g_k) \leq f(x_k - \alpha_k g_k), \\ x_k - \alpha_k g_k, & \text{otherwise,} \end{cases}$$

However, it was found that it is more efficient to add a positive constant to the objective function to make it nonnegative.

It is important to note that there is no line search in the SSC-SABB algorithm. The SSC-SABB algorithm switches between the two algorithms. All the switching is decided by an extra evaluation of the objective function value. This requires less computational work than an exact line search procedure and should be comparable to an inexact line search procedure. Since

the extra evaluation off can be done easily in parallel, it should not cause loss of speed in real computation.

It will be seen from Section 4 that, to a large extent, the supervisor guarantees the global convergence of the SSC-SABB algorithm, while the search engine decides the local convergence rate, at least when $\epsilon = 0$. As far as the supervisor is concerned, the gain is a possible increase of speed and efficiency; as far as the search engine is concerned, it may obtain extra robustness and global convergence.

4. Global Convergence and Convergence Rate of the SSC-SABB Algorithm

In this section, we examine the convergence and speed of the SSC-SABB algorithm. We will assume that $\epsilon = 0$ as mentioned in Section 1. The analysis carried out here should pave the way for fuller theoretical investigations on the algorithm. Due to the special selection of SR in SSC-SABB, the resulting algorithm is always globally convergent for a wide selections of $\{T_k\}$. In the SSC-SABB, algorithm, $\{T_k\}$ adjusts the balance between robustness and efficiency.

Theorem 4.1. Let f be twice continuously differentiable and bounded below. Assume that ∇f is Lipschitz with a global Lipschitz constant. Let $\{x_k\}$ be generated by the SSC-SABB algorithm defined in Section 3. Assume that $T_k \leq 1$, for $k = 0, 1, 2, \dots$. Then, there is an $\epsilon(f) > 0$ such that $\{\sum_0^k t_k |\nabla f(x_k)|^2\}$ is convergent as $k \rightarrow \infty$ for any sequence $\{t_k\}$ such that there is an $N > 0$ satisfying $t_k \leq \epsilon(f)$, for $k \geq N$.

Proof. For ease of exposition, we assume that $f \geq 0$. It follows from the definition of the algorithm that, for any $k \geq 0$,

$$\begin{aligned} f(x_{k+1}) &\leq \max(f(x_k - t_k g_k), T_k f(x_k - t_k g_k)) \\ &= \max(1, T_k) f(x_k - t_k g_k) \\ &\leq f(x_k - t_k g_k) \\ &\leq f(x_k) - t_k g_k^T g_k + t_k^2 g_k^T H_k g_k / 2, \end{aligned}$$

where H_k is the Hessian matrix of f at a point θ_k in the line segment $[x_k, x_{k+1}]$. Therefore, for $k \geq 1$,

$$f(x_{k+1}) \leq f(x_0) - \sum_0^k t_k |g_k|^2 + \sum_0^k C t_k^2 |g_k|^2,$$

where C depends only on f . Then, there is an $\epsilon > 0$ such that

$$t_k(1 - C t_k) \geq c' t_k, \quad \text{for } t_k \leq \epsilon,$$

where $c' > 0$ is a constant independent of k . Therefore, if there is a $N > 0$ such that $t_k \leq \epsilon$ for $k \geq N$, then $\sum_0^k t_k |g_k|^2$ is convergent as $k \rightarrow \infty$ as f is bounded below. \square

It follows from the above proof that the SSC-SABB algorithm is decreasing if $T_k \leq 1$ and k is large enough. Consequently, we have

$$\lim_{k \rightarrow \infty} f(x_k) = f(x^*),$$

where x^* is a stationary point of f if, for instance, the objective function has bounded level sets. Therefore, we have

$$\lim_{k \rightarrow \infty} x_k = x^*,$$

if, for example, the objective function is uniformly convex.

It seems that some convergence results can be established similarly for the case where the noises are deterministic; that is, f and g_k are only some approximation of F and ∇F . For instance, assume that ϵ is small in the function value evaluation, but significant in the gradient estimation, e.g., when using a finite difference approximation scheme to estimate ∇F . Then, it follows from the proof that, if g_k is only an approximation of $\nabla F(x_k)$, global convergence will still hold provided that

$$|g_k| \leq C|\nabla F(x_k)|, \quad (g_k, \nabla F(x_k)) \geq c|\nabla F(x_k)|^2,$$

where $c, C > 0$ are independent of k . These conditions are quite light and easy to meet. In fact, g_k is allowed to be far away from $\nabla F(x_k)$, e.g., twice as large as $\nabla F(x_k)$ with a 45 degree angle between them. This indicates the strong robustness of the SSC-SABB algorithm.

Global convergence may not hold when $T_k > 1, k = 0, 1, 2, \dots$, due to a weaker supervision. We construct here a 1-dimensional counterexample. For the one-dimensional case, we have that

$$s_k = b_k g_k = [g_k / (g_k - g_{k-1})] s_{k-1}. \quad (2)$$

For j sufficiently large, it is possible to choose the following steps and gradients:

$$\begin{aligned} s_{4j+1} &= 1, & s_{4j+2} &= (\sqrt{5} + 1)/2, & s_{4j+3} &= -1, & s_{4j+4} &= -(\sqrt{5} + 1)/2, \\ g_{4j+1} &= -1, & g_{4j+2} &= -(\sqrt{5} - 1)/2, & g_{4j+3} &= 1, & g_{4j+4} &= (\sqrt{5} - 1)/2. \end{aligned}$$

It follows that the above steps and gradients satisfy (2) and

$$g_{4j+i}^T s_{4j+1} < 0, \quad \sum_{i=1}^4 s_{4j+i} = 0.$$

Pick any value as \bar{x}_1 and let

$$\bar{x}_j = \bar{x} + \sum_{i=1}^j x_i.$$

Then, we construct a continuous function \tilde{f} such that

$$\tilde{f}(\bar{x}_i) \equiv 1 \text{ and } \nabla \tilde{f}(\bar{x}_i) = g_{4j+i}, \quad \text{for } i = 1, 2, 3, 4.$$

For this function, if k is large, if it happens that $x_k = \bar{x}_1$ and $x_{k+1} = \bar{x}_2$, and if $T_k \equiv T > 1$, the SSC-SABB algorithm will always use the search engine and hence will cycle between the four points \bar{x}_i , $i = 1, 2, 3, 4$. This counterexample actually tells us that such a cycle may happen in any case where the search engine is only locally convergent if $T_k \equiv T > 1$.

Actually, the case where sometimes $T_k > 1$ is important, because this could increase the efficiency of the resulting algorithm. There may be many different ways to cure this nonconvergence problem. For instance, let $S_k = \max(1, T_k)$. Then, it follows from the proof of the above theorem that the global convergence still holds provided $\prod_0^\infty S_k < \infty$. Therefore, one can always take a finite numbers of $T_k = T > 1$. Another simpler way to ensure global convergence is to let a finite numbers of $T_k = 1$, and then let the rest of $T_k = T > 1$. Global convergence can still be established, as it will be shown in the following theorem. We also examine the convergence rate of the SSC algorithm. For ease of exposition, we assume that $t_k \rightarrow 0$ in the following theorem, though it is not difficult to see that it holds for the case where t_k is very small after k is large enough, as in Theorem 4.1.

Theorem 4.2. Let $f > 0$ be a three times continuously differentiable function and strictly convex. Let x^* be the minimizer of f . Let $\{x_k^N\}$ be defined by the SSC-SABB algorithm of Section 3 with the following choice of $\{T_k\}$. Let $N > 0$ be a fixed integer and $T > 1$ a fixed real number. Let $T_k = 1$ for $k = 0, 1, 2, \dots, N$ and $T_k = T > 1$ for $k > N$. Then, there is a $N(x_0, f) > 0$ such that $\{x_k^N\}$ is convergent, whenever $N > N(x_0, f)$. Furthermore, the SSC-SABB algorithm is as fast as the BB algorithm locally, at least R-linearly convergent, whenever $N > N(x_0, f)$.

Proof. The proof of the above theorem could be made rather technical, as we have to first prove that the BB algorithm is locally R-linearly convergent. Although its local convergence has been proved for a quadratic objective function in Ref. 19, R-linear convergence or even simply convergence of the BB algorithm for general convex objective functions still needs many extra tedious estimates to prove. On the other hand, the principle of the proof is quite simple. Therefore, we skip some of the details for these estimates.

We show first the R-linear convergence of the BB algorithm, if x_k is sufficient close to x^* , where x_k is generated by the BB formula. For any k , given $\hat{x}_k = x_k$ and $\hat{x}_{k+1} = x_{k+1}$, we define $\{\hat{x}_{k+j} : j = 0, 1, 2, \dots\}$ to be the iterations generated by the BB algorithm for the quadratic function

$$\hat{f}(x) = f(x^*) + (1/2)(x - x^*)^T H(x - x^*),$$

where H is the Hessian matrix of f at x^* . It follows from Ref. 19 that $\{\hat{x}_{k+j}\}$ converges to x^* as $j \rightarrow \infty$, if x_k is very near x^* .

Then, for any $1 \leq l \leq m$, where m is some fixed integer, if

$$|\hat{x}_{k+j} - x^*| \geq c_1 |\hat{x}_k - x^*|, \quad j = 1, \dots, l,$$

where $c_1 > 0$ is constant, we can prove by induction that there exists a positive constant $c_2 > 0 = c_2(m, c_1, H)$, independent of k , such that

$$|x_{k+j} - \hat{x}_{k+j}| \leq c_2 |x_k - x^*|^2, \quad j = 1, \dots, l. \quad (3)$$

Furthermore, it can be shown (see Ref. 21) that there exist a constant $c_3 \in (0, 1)$ and an integer m which depends only on c_3 and H such that, for any $k \geq 2$, there exists an integer $l \in [1, m]$ such that

$$|\hat{x}_{k+l} - x^*| \leq c_3 |\hat{x}_k - x^*|. \quad (4)$$

Now, let $\delta = (1 - c_3)/2c_2$ and let k_0 be so large that

$$|x_{k_0} - x^*| \leq \delta. \quad (5)$$

For this k_0 , let $\hat{x}_{k_0} = x_{k_0}$ and $\hat{x}_{k_0+1} = x_{k_0+1}$, and denote by k_1 the least index for which

$$|\hat{x}_{k_1} - x^*| \leq c_3 |\hat{x}_{k_0} - x^*|. \quad (6)$$

It is obvious that $k_1 - k_0 \leq m$. Then, by (3), (6), (5), and the choice of δ , we can show that

$$|x_{k_1} - x^*| \leq |\hat{x}_{k_1} - x^*| + |x_{k_1} - \hat{x}_{k_1}| \leq c_4 |x_{k_0} - x^*|,$$

where

$$c_4 = (1 + c_3)/2 < 1.$$

Repeating this procedure, we can then obtain an infinite subsequence $\{k_i : i = 1, 2, \dots\}$ such that

$$k_{i+1} - k_i \leq m \text{ and } |x_{k_{i+1}} - x^*| \leq c_4 |x_{k_i} - x^*|, \quad i = 1, 2, \dots \quad (7)$$

In addition, note that there exists a constant $c_5 > 0$ such that the relation

$$|x_{k+1} - x^*| \leq c_5 |x_k - x^*| \quad (8)$$

holds for any large k . By (7) and (8), then we can prove that

$$|x_{k_0+j} - x^*| \leq M c_6^j |x_{k_0} - x^*|,$$

where

$$M = c_4^{-1} c_5^{m-1} \quad \text{and} \quad c_6^{1/m} = c_4^{1/m} < 1.$$

The above relation shows that the BB algorithm is R-linearly convergent.

In the following, we show that the SSC-SABB algorithm will take only the BB stepsizes when it starts from a point very near the minimizer x^* ; then, we prove that there is a subsequence of $\{x_k^N\}$ very close to x^* provided N is chosen large enough.

As $f(x^*) > 0$, there is a $r_0 > 0$ such that

$$Tf(x - t_k \nabla f(x)) > f(y), \quad \forall k \geq 0,$$

as long as

$$|x - x^*| < r_0 \quad \text{and} \quad |y - x^*| < r_0,$$

since $T > 1$ and $\{t_k\}$ is bounded.

Now let $\{y_k\}$, $k = 0, 1, 2, \dots$, be generated by the SSC-SABB algorithm with $y_0 = x_0$ and $T_k \equiv 1$. Then, from Theorem 4.1, there will be a $k_0 > 0$, depending on x_0 and f , such that

$$|y_{k_0} - x^*| < \min(\bar{r}_0, \delta)/M,$$

where $\bar{r}_0 \leq r_0$ and δ is defined above, since f is strictly convex. Let the sequence $\{x_k^N\}$ be generated via the SSC-SABB algorithm from x_0 by letting $T_k = 1$ for $k = 1, 2, \dots, N$ and $T_k = T > 1$ for $k = N + 1, \dots$. Let $N_0 = k_0$. Note that the first N_0 elements $\{x_0^N, x_1^N, \dots, x_{N_0}^N\}$ of this sequence are identical with $\{y_0, y_1, \dots, y_{N_0}\}$ provided $N \geq N_0$. Let x_{k_0+i} , $i = 1, 2, \dots$, be the sequence generated by the BB algorithm from $x_{k_0}^{N_0} = x_{N_0}^{N_0} = x_{N_0}^N$, $N \geq N_0$. Clearly, \bar{r}_0 can be made so small that

$$|x_{k_0+1} - x^*| < \min(r_0, \delta)/M.$$

Therefore,

$$|x_{k_0+i} - x^*| \leq M c_6^i |x_{k_0}^{N_0} - x^*| < \min(r_0, \delta), \quad i = 1, 2, \dots,$$

Hence, the SSC-SABB algorithm will use only the BB stepsizes after $k_0 = N_0$ according to its switching rule. Therefore, the sequence $\{x_k^N\}$, generated via the SSC-SABB algorithm by letting $T_k = 1$ for $k = 1, 2, \dots, N$ and $T_k = T > 1$ for $k = N + 1, \dots$, is as fast as the BB algorithm, at least R-linearly convergent for any $N \geq N_0$.

In practical computation, T_k is often fixed to a constant $T > 1$. In general, the convergence rate of the SSC-SABB algorithm is much better than

that of the SA algorithm, due to the faster search engine used in the algorithm (the BB algorithm is sometimes R-superlinear). Indeed, this is confirmed in our numerical tests. The condition $f > 0$ may be met by adding a large positive number C to the original objective function.

The above analysis confirms our expectations, discussed in Section 2, that is, that the global convergence is largely decided by the supervisor SR, while efficiency of the algorithm depends much on the search engine SE. In Section 5, we carry out some numerical tests for the SSC-SABB algorithm.

5. Numerical Tests

We present some numerical experiments for both deterministic and stochastic test problems. The purpose of these tests is to see whether or not the SSC algorithm is efficient in the lower noise case and robust in the stochastic case.

5.1. Deterministic Case, ϵ Negligible. We use 22 test problems in the deterministic experiments. Problems 1–18 are drawn from Moré, Garbow, and Hillstom (Ref. 22), which are well known and certainly not trivial; most of them are not convex. Problems 19–20 are stated in the Raydan paper (Ref. 20). We adopt the initial values used in the above works.

Problems 21–22 are described as follows:

$$\begin{aligned} \text{(P21)} \quad & f(x) = x^4 + x^2 + 100, x \in R^1, \quad \text{with } x_0 = 10, \\ \text{(P22)} \quad & f(x) = \sum_{i=1}^{50} ix_i^2 + \sum_{i=1}^{49} x_i x_{i+1}, \quad \text{with } x_0 = (1, 1, \dots, 1)^T. \end{aligned}$$

Problems 19–22 represent good or normal ones.

We compare Algorithm SSC-SABB with Algorithms SA and GBB. The latter was chosen because it was reported to be rather efficient and because it uses the BB stepsize as well. For GBB, we adopt all the recommended restrictions and procedures given in Ref. 20. For SSC-SABB, we have no restriction for α_k . We take $T_k = 5$ in all the tests. We also tried $T_k = 1$ for a few initial steps and then $T_k = 5$. However, there is not much difference. We have used three different sequences of $\{t_k\}$ in our experiments:

$$t_k = \min(1.5/k, 0.01), \quad t_k = \min(1.5/\sqrt{k}, 0.01), \quad t_k = 0.01.$$

In Table 1 and 2, we report the number of iterations and number of function evaluations, with the stopping rule

$$|\nabla f(x_k)| \leq 10^{-6}.$$

Table 1. Comparing the GBB and SSC-SABB algorithms, number of iterations.

P	n	GBB	$\min(1.5/k, 0.01)$	$\min(1.5/\sqrt{k}, 0.01)$	0.01
1	3	221	143	143	143
2	6	1073	45	45	45
3	3	4	5	5	5
4	2	> 9999	> 9999	> 9999	> 9999
5	3	266	14	14	14
6	6	12	19	19	19
7	9	> 9999	> 9999	> 9999	> 9999
8	8	145	> 9999	> 9999	> 9999
9	3	14	20	20	20
10	2	> 9999	> 9999	> 9999	> 9999
11	4	50	128	128	128
12	3	> 9999	1	1	1
13	20	82	92	92	92
14	14	75	415	289	289
15	16	> 9999	402	320	320
16	2	44	48	48	48
17	4	914	668	1573	1573
18	8	56	68	68	68
19	100	7	7	7	7
20	100	105	104	104	104
21	1	14	17	17	17
22	50	113	101	101	101

Let us note that the number of gradient evaluations is just the number of iterations plus one. The maximum number of function evaluations is set to 9999. It is found that SA fails for most test problems, so that it is not included here.

From Tables 1–2, we see that, on average, SSC-SABB seems to be as efficient as GBB in terms of the number of function and gradient evaluations. It was found that SSC-SABB is faster than GBB in terms of CPU time: SSC solves the commonly solved problems in 0.42 s, while GBB uses 0.78 s on a Sun UltraSparc 1 station. The GNU g77 compiler was used without any optimization flags. This may be caused by the fact that, in two cases, SSC finds different local minima, thus consuming less CPU time. Also, the global GBB parameter setting may not be optimal for individual problems. However, we are able to state that the overall performances of these two algorithms are similar.

From Tables 1–2, one can also see that the performance of the SSC is not very sensitive to the selection of $\{t_k\}$.

In the deterministic case, it is not difficult to speed up SSC-SABB. One can use, for example, the following rule: if $|g_k| \leq 0.01$, then only the

Table 2. Comparing the GBB and SSC-SABB algorithms, number of function evaluations.

P	n	GBB	$\min(1.5/k, 0.01)$	$\min(1.5/\sqrt{k}, 0.01)$	0.01
1	3	272	287	287	287
2	6	1458	91	91	91
3	3	6	11	11	11
4	2	> 9999	> 9999	> 9999	> 9999
5	3	352	29	29	29
6	6	18	39	39	39
7	9	> 9999	> 9999	> 9999	> 9999
8	8	150	> 9999	> 9999	> 9999
9	3	17	41	41	41
10	2	> 9999	> 9999	> 9999	> 9999
11	4	61	257	257	257
12	3	> 9999	3	3	3
13	20	87	185	185	185
14	14	107	831	579	579
15	16	> 9999	805	641	641
16	2	51	97	97	97
17	4	1238	1337	3147	3147
18	8	67	137	137	137
19	100	8	15	15	15
20	100	119	209	209	209
21	1	18	35	35	35
22	50	128	203	203	203

BB step is used. Furthermore, we can use a line search in SE. The resulting algorithm will be referred to as SAGBB; that is, we use SA with $t_k = \min(1.5/\sqrt{k}, 0.01)$ as supervisor and the GBB algorithm as search engine. The test results with these improvements are shown in Tables 3 and 4.

It can be seen from Tables 3–4 that the speed of SSC-SABB is further increased. Particularly, SAGBB solves almost all the test problems. In fact, it needs only an extra few tens of iterations to solve the only unsolved test problem (Problem 7).

5.2. Stochastic Case. The purpose of our experiments here is to test the robustness of SSC-SABB. We still use the above 22 problems as the underlying exact models, but add the noise $\epsilon = 0.1 N(0, 1)$, where $N(0, 1)$ is the standard normal random variable. Therefore, the test setting reads

$$\min[F(x) + \epsilon].$$

Table 3. Accelerating the SSC-SABB algorithm, number of iterations.

P	n	GBB	$\min(1.5/k, 0.01)$	$\min(1.5/\sqrt{k}, 0.01)$	0.01
1	3	112	112	112	128
2	6	45	45	45	76
3	3	5	5	5	5
4	2	> 9999	> 9999	> 9999	4
5	3	14	14	14	20
6	6	19	19	19	19
7	9	> 9999	> 9999	> 9999	> 9999
8	8	32	32	32	31
9	3	20	20	20	18
10	2	> 9999	> 9999	> 9999	174
11	4	128	128	128	130
12	3	1	1	1	1
13	20	192	192	192	535
14	14	416	266	266	61
15	16	261	214	214	546
16	2	36	36	36	33
17	4	564	1537	1537	659
18	8	68	68	68	75
19	100	7	7	7	7
20	100	104	104	104	83
21	1	17	17	17	17
22	50	101	101	101	117

We then use $F(x) + \epsilon$ to evaluate the function value and use $\nabla[F(x) + \epsilon]$ to estimate the gradient. It is clear that the noise can be dominant in the gradient estimation when the approximation is near a solution, where $\nabla F = 0$. In fact, most noisy optimization problems are certainly not trivial to solve.

We compare SSC-SABB, SA, and GBB with 100 restarts at each iteration. We use the following stopping rule:

$$\begin{aligned} |\hat{x}_k - x^*| &< \text{eps}, & \text{if } x^* \text{ is known,} \\ |(\hat{f}_k - F(x^*))| &< (\text{eps})^2, & \text{otherwise,} \end{aligned}$$

where $\text{eps} = 0.01$, x^* is the minimizer, and

$$\hat{x}_k = \sum_{i=0}^{19} x_{k-i}/20, \quad \hat{f}_k = \sum_{i=0}^{19} f(x_{k-i})/20.$$

We test SA with $t_k = C/(k+1)$ and $t_k = 0.001$, GBB with 100 restarts, and SSC-SABB with $T_k = 1.0$ and $t_k = C/(k+1)$, where in contrast to the deterministic case, the value of C has to be adjusted according to the problem: $C = 0.1$ for most test problems; $C = 1.0$ for Problems 19–23; $C = 0.001$ for Problems 3, 12, 17–18; $C = 0.000001$ for Problems 4, 6, 10, 11; Although

Table 4. Accelerating the SSC-SABB algorithm, number of function evaluations.

P	n	GBB	$\min(1.5/k, 0.01)$	$\min(1.5/\sqrt{k}, 0.01)$	0.01
1	3	209	209	209	278
2	6	74	74	74	123
3	3	7	7	7	7
4	2	> 9999	> 9999	> 9999	16
5	3	25	25	25	49
6	6	38	38	38	38
7	9	> 9999	> 9999	> 9999	> 9999
8	8	51	51	51	53
9	3	36	36	36	37
10	2	> 9999	> 9999	> 9999	608
11	4	248	248	248	250
12	3	3	3	3	3
13	20	249	249	249	648
14	14	828	529	529	133
15	16	412	356	356	823
16	2	63	63	63	67
17	4	1092	2957	2957	1357
18	8	117	117	117	140
19	100	13	13	13	13
20	100	151	151	151	132
21	1	33	33	33	35
22	50	154	154	154	178

SSC-SABB is not very sensitive to the selection of $\{t_k\}$, the starting value of t_0 is very important to ensure convergence, due to the stochastic nature of the test problems. We emphasize again that these stochastic problems are nontrivial and that the above adjustments are commonly used in engineering computation in order to solve realistic problems.

The results are presented in Table 5, where:

HNMLS is GBB with 100 restarts;

SSC-SABB is similar to the SSC-SABB used in the deterministic case with certain restrictions on the stepsizes (see Ref. 20);

SSC-SABB1 is a modified version of SSC-SABB; for the first 10 iterations, it uses the nonmonotone line search with 10 restarts; if the line search fails, it switches to SA steps.

Each test problem is given 10 runs, with different seed for the random number generator in each one of them. In Table 5, the number of successful runs is reported for these algorithms.

It was found that the SSC algorithm solves many more problems than SA and GBB with restarts. In fact, there seems to be no existing algorithm

Table 5. Comparing the SA, GBB, and SSC-SABB algorithms, stochastic case.

P	SA ($t_k = C/(k+1)$)	SA ($t_k = 0.001$)	HNMLS	SSC-SABB	SSC-SABB1
1	0	0	0	10	8
2	0	0	0	1	9
3	10	0	0	10	9
4	0	0	0	0	0
5	0	4	1	10	9
6	0	10	8	10	10
7	0	1	1	9	10
8	0	0	0	2	9
9	0	0	0	10	10
10	0	0	0	0	0
11	0	0	0	10	10
12	0	0	0	0	0
13	0	0	0	5	8
14	0	0	0	0	0
15	0	0	0	0	0
16	0	9	10	10	10
17	0	0	0	0	0
18	10	0	0	10	10
19	0	0	0	0	0
20	0	0	0	0	0
21	0	10	10	10	10
22	0	10	10	10	10

that can solve so many difficult noisy optimization problems. It is found that increasing the maximum number of function evaluations (currently 9999) helps the SSC algorithms to solve even more problems, while this makes little difference to the other algorithms. Central difference approximations have been used also to estimate the gradient and similar results were observed. These results will be reported elsewhere.

We also analyzed the speed of the tested algorithms for the last quadratic problem in Table 6. It can be seen that SSC-SABB is much faster than SA in this classical test problem. We also tested a quadratic problem where

Table 6. Speed of the SA, GBB, and SSC-SABB algorithms, quadratic functions.

Algorithms	Number of function evaluations	Number of gradient evaluations	CPU (sec)
SA, $t_k = 0.1/(k+1)$	> 9999	> 9999	—
SA, $t_k = 0.1/\sqrt{k+1}$	> 9999	> 9999	—
SA, $t_k = 0.001$	2207	2207	128
GBB, 100 restarts	Line search failure	Line search failure	—
SSC-SABB, n = 50	183	366	4
SSC-SABB, n = 100	230	460	15

stochastic noises are present in the coefficients, and similar results were observed, though GBB and SA show improved performance since, in that particular problem, the noises are diminishing when the approximation x_k is approaching the real solution.

We also tested a range of different values of T_k in algorithm SSC-SABB. In all the tests, T_k was kept as a constant between 1 and 10. In the deterministic case, it was found that for the good Problems 19–22 and for Problems 3, 6, 9, the values of T_k do not seem to affect much the speed of SSC-SABB as long as one keeps $T_k \geq 1$. For the other cases, we observed a significant reduction in the iteration numbers needed to solve these problems via SSC-SABB, when increasing the values of T_k from 1 toward 3. Clearly, this is because the algorithm now used the search engine more frequently. Then, such reduction was hardly observed when the values of T_k were taken in the interval [4, 6]. Further increase in the values may actually increase the iteration numbers needed to solve these problems via SSC-SABB. This is probably due to the local convergence nature of the search engine. In summary, it seems that one should take values of T_k in the interval [4, 6] in the first place.

For the stochastic case, $T_k = 1$ or nearly 1 appears to be always a good choice. This value seems to give a good balance between speed and robustness.

6. Conclusions

We investigated a general framework, that of supervisor-searcher co-operation (SSC), which seems to offer a promising way to incorporate appealing attributes into the design of new optimization algorithms. A new class of algorithms was proposed within this framework, and a particular example from this class was examined thoroughly. Our theoretical results suggest that SSC exhibits the desirable features of its parent algorithms. Furthermore, our numerical tests indicate that the algorithm is both efficient in deterministic problems and robust in stochastic problems. The fact that neither a line search nor fixed stepsizes are used may be the cause of this desirable behavior. The computational work needed is also very light. In a forthcoming paper, we shall examine other examples from this class of algorithms.

References

1. DENNIS, J. E., and SCHNABEL, R. B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.
2. FLETCHER, R., *Practical Methods of Optimization*, 2nd Edition, John Wiley and Sons, Chichester, England, 1987.

3. GRIPPO, L., LAMPARIELLO, F., and LUCIDI, S., *A Nonmonotone Line Search Technique for Newton's Methods*, SIAM Journal on Numerical Analysis, Vol. 23, pp. 707–716, 1986.
4. KIEFER, J., and WOLFOWITZ, J., *Stochastic Estimation of the Maximum of a Regression Function*, Annals of Mathematical Statistics, Vol. 23, pp. 462–466, 1952.
5. ROBBINS, H., and MONRO, S., *A Stochastic Approximation Method*, Annals of Mathematical Statistics, Vol. 22, pp. 400–407, 1951.
6. BENVENISTE, M., METIVIER, M., and PRIOURET, P., *Adaptive Algorithms and Stochastic Approximation*, Springer Verlag, Berlin, Germany, 1990.
7. KUSHNER, H. J., and CLARK, D. C., *Stochastic Approximation Methods for Constrained and Unconstrained Systems*, Springer, New York, NY, 1978.
8. KUSHNER, H. J., and YIN, G. G., *Stochastic Approximation Algorithms and Applications*, Springer, New York, NY, 1997.
9. LJUNG, L., *System Identification Theory for the User*, Prentice-Hall, Englewood Cliffs, New Jersey, 1986.
10. URYAS'EV, S. P., *A Stochastic Quasigradient Algorithm with Variable Metric*, Annals of Operations Research, Vol. 39, pp. 251–267, 1992.
11. ELSTER, C., and NEUMAIER, A., *A Trust-Region Method for the Optimization of Noisy Functions*, Computing, Vol. 58, pp. 31–46, 1997.
12. YAN, D., and MUKAI, H., *Optimization Algorithm with Probabilistic Estimation*, Journal of Optimization Theory and Applications, Vol. 79, pp. 345–371, 1993.
13. CONN, A. R., SCHEINBERG, K., and TOINT, P. L., *Recent Progress in Unconstrained Nonlinear Optimization without Derivatives*, Mathematical Programming, Vol. 79, pp. 397–414, 1997.
14. POWELL, M. J. D., *Direct Search Algorithms for Optimization Calculations*, Acta Numerica, pp. 287–336, 1998.
15. ANDERSON, E. J., and FERRIS, M. C., *A Direct Search Algorithms for Optimization with Noisy Function Evaluations*, Working Paper 97-010, Australian Graduate School of Management, 1997.
16. BARTON, R. R., and IVEY, J. S., *Nelder-Mead Simplex Modification for Simulation Optimization*, Management Science, Vol. 42, pp. 954–972, 1996.
17. FU, M. C., *Optimization via Simulation: A Review*, Annals of Operations Research, Vol. 53, pp. 199–247, 1994.
18. BARZILAI, J., and BORWEIN, M., *Two-Point Step Size Gradient Methods*, IMA Journal of Numerical Analysis, Vol. 8, pp. 141–148, 1988.
19. RAYDAN, M., *On the BB Choice of Steplength for the Gradient Method*, IMA Journal of Numerical Analysis, Vol. 13, pp. 321–326, 1993.
20. RAYDAN, M., *The Barzilai and Borwein Gradient Method for the Large-Scale Unconstrained Minimization Problem*, SIAM Journal on Optimization, Vol. 7, pp. 26–33, 1997.
21. DAI Y. H., and LIAO L. Z., *R-Linear Convergence of the Barzilai and Borwein Gradient Method*, Report ICM-99-039, Institute of Computational Mathematics and Scientific Computing, Beijing, China, 1999.
22. MORÉ, J., GARBOW, B. S., and HILLSTROM, K. E., *Testing Unconstrained Optimization Software*, ACM Transactions on Mathematical Software, Vol. 7, pp. 17–41, 1981.