PERGAMON Computers and Mathematics with Applications 37 (1999) 73–88

An International Journal
computers & mathematics
with applications

Space-Decomposition Minimization Method for Large-Scale Minimization Problems

CHIN-SUNG LIU
Research Assistant of Applied Optimum Design Laboratory
Department of Mechanical Engineering, National Chiao Tung University
Hsinchu 30050, Taiwan, R.O.C.

CHING-HUNG TSENG
Professor of Applied Optimum Design Laboratory
Department of Mechanical Engineering, National Chiao Tung University
Hsinchu 30050, Taiwan, R.O.C.

(Received August 1997; accepted September 1998)

Abstract—This paper introduces a set of new algorithms, called the Space-Decomposition Minimization (SDM) algorithms, that decomposes the minimization problem into subproblems. If the decomposed-space subproblems are not coupled to each other, they can be solved independently with any convergent algorithm; otherwise, iterative algorithms presented in this paper can be used. Furthermore, if the design space is further decomposed into one-dimensional decomposed spaces, the solution can be found directly using one-dimensional search methods. A hybrid algorithm that yields the benefits of the SDM algorithm and the conjugate gradient method is also given.

An example that demonstrates application of SDM algorithm to the learning of a single-layer perceptron neural network is presented, and five large-scale numerical problems are used to test the SDM algorithms. The results obtained are compared with results from the conjugate gradient method.

Keywords—Unconstrained minimization, Decomposition method, Direct-search method, Large-scale problem.

1. INTRODUCTION

This paper introduces a set of space-decomposition minimization (SDM) algorithms for solving the unconstrained minimization problem

$$\min_{x \in \Re^n} f(x), \tag{1}$$

where $f : \Re^n \to \Re$ is a lower bounded and continuously differentiable function.

The space-decomposition minimization (SDM) algorithms are based on decomposing the design space $S \in \Re^n$ into individual subspaces. The minimization problem can then be decomposed into q subproblems and the final minimization solution is the combination of the q decomposed-space minimization solutions.

The research reported in this paper was supported under a project sponsored by the National Science Council Grant, Taiwan, R.O.C., NSC 85-2732-E-009 -010.

0898-1221/99/$ - see front matter Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX
PII: S0898-1221(99)00088-7

The space-decomposition minimization (SDM) algorithms can be considered as extensions of the Parallel Variable Distribution (PVD) algorithm and associated decomposition methods. The PVD algorithm, proposed by Ferris and Mangasarian [1], and extended by Solodov [2], is a method that decomposes the design variable x into q blocks $x_1, \ldots, x_q$ and distributes them among q processors, where $x_\ell \in \Re^{n^\ell}$ with $\sum_{\ell=1}^{q} n^\ell = n$. Mangasarian [3] also introduced a method that assigns a portion of the gradient $\nabla f(x)$ to q processors. Similarly, the n-dimension real-space $\Re^n$ is also decomposed into q n^ℓ-dimension subspaces.

Decomposition methods for decomposing the minimization problem into subproblems have been proposed by several researchers. Kibardin [4] decomposed minimization problem (1) into $f(x) = \sum_{i=1}^{N} f_i(x)$, where $x \in \Re^n$ and $f_i(x)$ is a convex function. Mouallif, Nguyen and Strodiot [5] also decomposed minimization problem (1) into, where $f(x) = f_0(x) + \sum_{i=1}^{N} f_i(x)$, where $x \in \Re^n$, f_0 is a differentiable, strongly convex function, and $f_i(x)$ is a convex function. In these studies, the computing efficiency was shown to increase when the original minimization problem could be decomposed into subproblems.

In this paper, the PVD algorithm and the decomposition methods are combined into the space-decomposition minimization (SDM) algorithms. It is shown that any convergent algorithm that satisfies the descent condition can be applied to solve the decomposed-space subproblems. In addition, if the design space is further decomposed into one-dimensional decomposed spaces, one-dimensional search methods can be used directly to solve the subproblems. That is, the SDM algorithm can be considered a direct search method.

Direct-search methods search for minimum solutions directly without requiring analytic gradient information. The multiple mutually conjugate search directions are commonly used in the direct search methods [6–8]. Line search along these multiple search directions can be evaluated simultaneously on different computers using only evaluation of minimization function. In this paper, the special pseudo-conjugate directions [9] that parallel the coordinate axes are used for the direct-search SDM algorithm, and one-dimensional search methods, including exact and nonexact methods, can be used directly to solve the one-dimensional subproblems.

This paper is organized as follows. The space-decomposition minimization (SDM) algorithm is presented in Section 2. The direct-search space-decomposition minimization (direct-search SDM) algorithm is presented in Section 3. In Section 4, a hybrid SDM algorithm that combines the algorithms presented in Sections 2 and 3 is introduced. An example of application to the neural networks is given in Section 5, and numerical experiment results are given in Section 6. The conclusions along with further research directions are given in Section 7, and all test problems are listed in the Appendix.

The notation and terminology used in this paper are described below [3].

$S \in \Re^n$ denotes n-dimensional Euclidean design space, with ordinary inner product and associated two-norm $\| \bullet \|$. Italic characters denoting variables and vectors. For simplicity of notation, all vectors are column vectors and changes in the ordering of design variables are allowed throughout this paper. Thus, the design variable vector $x \in S$ can be decomposed into subvectors. That is, $x = [x_{S_1}, \ldots, x_{S_q}]^\top$, where x_{S_i} are subvector or subcomponent of x.

2. SPACE-DECOMPOSITION MINIMIZATION (SDM) ALGORITHM

The space-decomposition minimization algorithm is based on the nonoverlapping decomposed-space set and the decomposed-space minimization function defined below.

DEFINITION 2.1. NONOVERLAPPING DECOMPOSED-SPACE SET. *For the design space S spanned by $\{x \mid x \in \Re^n\}$, if the design variable x is decomposed into $x = [x_{S_1}, \ldots, x_{S_q}]^\top$, then the decomposed space S_i spanned by $\{x_{Si} \mid x_{Si} \in \Re^{n_i}$, where $\sum_{i=1}^{q} n_i = n\}$ forms a nonoverlapping*

decomposed-space set $\{S_1, \ldots, S_q\}$. *That is,*

$$\bigcup_{i=1}^{q} S_i = S \quad \text{and} \quad S_i \cap S_j = \emptyset, \qquad \text{if } i \neq j.$$

Moreover, the complement space of S_i is defined as

$$\bar{S}_i \text{ spanned by } \left\{ x_{\bar{S}_i} \mid \text{where } \left[x_{S_i}, x_{\bar{S}_i} \right]^\top = x \right\}. \tag{2}$$

That is, $\bar{S}_i \cup S_i = S$ and $\bar{S}_i \cap S_i = \emptyset$, $\forall i \in (1, q)$.

According to definitions of the nonoverlapping decomposed-space set, the minimization function (1) can be decomposed as

$$f(x) = f_{S_i}\left(x_{S_i}, x_{\bar{S}_i}\right) + f_{\bar{S}_i}\left(x_{\bar{S}_i}\right). \tag{3}$$

where $f_{S_i}(x_{S_i}, x_{\bar{S}_i})$ is the decomposed-space minimization function and $f_{\bar{S}_i}(x_{\bar{S}_i})$ is the complement decomposed-space minimization function.

COROLLARY 2.2. *According to (3), the complement decomposed-space minimization function* $f_{\bar{S}_i}(x_{\bar{S}_i})$ *is only a function of* $x_{\bar{S}_i}$. *That is, if* $x_{\bar{S}_i}$ *is a constant vector,* $f_{\bar{S}_i}(x_{\bar{S}_i})$ *can be treated as a constant value that can be removed during the process of minimizing decomposed-space subproblems.*

The following example uses the Powell test function listed in the Appendix to describe Definition 2.1 and Corollary 2.2.

EXAMPLE 2.3. The Powell minimization problem with four design variables is [9–11].

$$\min_{x \in \Re^4} f(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4.$$

Definition 2.1 and Corollary 2.2 suggest the nonoverlapping decomposed spaces S_i, $i = 1, \ldots, 4$, are spanned by $\{x_1\}$, $\{x_2\}$, $\{x_3\}$, $\{x_4\}$, and the complementary spaces $\bar{S}_i$, $i = 1, \ldots, 4$, are spanned by $\{x_2, x_3, x_4\}$, $\{x_1, x_3, x_4\}$, $\{x_1, x_2, x_4\}$, $\{x_1, x_2, x_3\}$, respectively. Thus, the decomposed-space minimization function f_{S_i} can be expressed by eliminating the component of $f(x)$ defined only in $\bar{S}_i$. That is,

$$\begin{aligned}
f_{S_1}\left(x_{S_1}, x_{\bar{S}_1}\right) &= (x_1 + 10x_2)^2 + 10(x_1 - x_4)^4, \\
f_{S_2}\left(x_{S_2}, x_{\bar{S}_2}\right) &= (x_1 + 10x_2)^2 + (x_2 - 2x_3)^4, \\
f_{S_3}\left(x_{S_3}, x_{\bar{S}_3}\right) &= 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4, \\
f_{S_4}\left(x_{S_4}, x_{\bar{S}_4}\right) &= 5(x_3 - x_4)^2 + 10(x_1 - x_4)^4.
\end{aligned}$$

These subproblems can then be solved using the methods presented in this paper. ∎

The definition of the nonoverlapping decomposed-space set leads to the following theorem.

THEOREM 2.4. UNCOUPLED SPACE-DECOMPOSITION THEOREM. *If minimization problem (1) can be decomposed into*

$$f(x) = \sum_{i=1}^{q} f_{S_i}\left(x_{S_i}\right), \tag{4}$$

where $\{S_1, \ldots, S_q\}$ *is the nonoverlapping decomposed-space set, then the minimization problem (1) can be decomposed into* q *uncoupled subproblems that can be solved independently using any convergent algorithm. That is,*

$$x^*_{S_i} \in \arg\min f_{S_i}(x_{S_i}), \qquad \forall i \in (1, q), \tag{5}$$

and the final minimization or stationary solution can be obtained from

$$x^* = \left[x^*_{S_1}, \ldots, x^*_{S_q}\right]^\top. \tag{6}$$

PROOF. From (4), it gets

$$\frac{\partial f(x)}{\partial x_j} = \frac{\partial f_{S_i}(x_{S_i})}{\partial x_j}, \qquad \forall x_j \in S_i. \tag{7}$$

Thus, from Definition 2.1 and (7), it can be concluded that

$$\frac{\partial f(x)}{\partial x_j} = \frac{\partial f_{S_i}(x_{S_i})}{\partial x_j}, \qquad \forall x_j \in S. \tag{8}$$

Since $x^*_{S_i}$ is the minimization solution for decomposed space S_i, it follows that

$$\frac{\partial f_{S_i}\left(x^*_{S_i}\right)}{\partial x_j} = 0, \qquad \forall x_j \in S_i. \tag{9}$$

Thus, from (8) and (9), it can be concluded that

$$\frac{\partial f\left(x^*\right)}{\partial x_j} = 0, \qquad \forall x_j \in S.$$

That is, $\|\nabla f(x^*)\| = 0$, and this is the minimization or stationary condition for minimization problem (1). ∎

Theorem 2.4 is efficient for some simple problems, and can save lots of processor time and memory resources. However, most minimization problems cannot be decomposed into uncoupled decomposed-space subproblems, so iterative algorithms must be used.

Before stating and proving the iterative space-decomposition minimization algorithm, the definition of forcing function [1,3] is required and is defined below.

DEFINITION 2.5. FORCING FUNCTION. *A continuous function σ from the nonnegative real line into itself is said to be a forcing function on the nonnegative real-number sequence $\{\xi_i\}$ if*

$$\sigma(0) = 0, \quad \text{with } \{\sigma(\xi)\} > 0 \text{ for } \xi > 0. \tag{10}$$

In addition, $\sigma(\xi_i) \to 0$ implies $\{\xi_i\} \to 0$.

The forcing function definition leads to the following space-decomposition theorem and proof.

THEOREM 2.6. SPACE-DECOMPOSITION THEOREM. *If the design space $S \in \Re^n$ of minimization problem (1) is decomposed into a nonoverlapping decomposed-space set $\{S_1, \ldots, S_q\}$, then minimization problem (1) can be solved iteratively in the q decomposed spaces using any convergent algorithm that satisfies the descent condition [3]:*

$$-\nabla f_{S_i}\left(x^k_{S_i}, x_{\bar{S}_i}\right)^\top d^k_{S_i} \geq \sigma_1\left(\left\|\nabla f_{S_i}\left(x^k_{S_i}, x_{\bar{S}_i}\right)\right\|\right) \tag{11}$$

and

$$f_{S_i}\left(x^k_{S_i}, x_{\bar{S}_i}\right) - f_{S_i}\left(x^{k+1}_{S_i}, x_{\bar{S}_i}\right) \geq \sigma_2\left(-\nabla f_{S_i}\left(x^k_{S_i}, x_{\bar{S}_i}\right)^\top d^k_{S_i}\right) \geq 0, \tag{12}$$

where $d^k_{S_i}$ is the search direction in decomposed-space S_i, σ_1 is a forcing function on the sequence $\{\|\nabla f_{S_i}(x^k_{S_i}, x_{\bar{S}_i})\|\}$, σ_2 is a forcing function on the sequence $\{-\nabla f_{S_i}(x^k_{S_i}, x_{\bar{S}_i})^\top d^k_{S_i}\}$, and $[x^{k+1}_{S_i}, x_{\bar{S}_i}]^\top = [x^k_{S_i}, x_{\bar{S}_i}]^\top + \alpha_k\, d^k_{S_i}$ in space S_i.

The final minimization or stationary solution can be obtained by directly combining the solutions of the q subproblems. That is, $x^* = [x^*_{S_1}, \ldots, x^*_{S_q}]^\top$.

PROOF. Using (12) and the forcing function definition yields

$$f_{S_i}\left(x_{S_i}^{k+1}, x_{\bar{S}_i}\right) \leq f_{S_i}\left(x_{S_i}^{k}, x_{\bar{S}_i}\right). \tag{13}$$

Expanding (13) using the Taylor series about point $x_{S_i}^k$ yields

$$f_{S_i}\left(x_{S_i}^{k}, x_{\bar{S}_i}\right) + \alpha_k \left(c_{S_i}^{k} \cdot d_{S_i}^{k}\right) \leq f_{S_i}\left(x_{S_i}^{k}, x_{\bar{S}_i}\right), \tag{14}$$

where $c_{S_i}^k$ is the gradient of $f_{S_i}(x_{S_i}^k, x_{\bar{S}_i})$ in space S_i. Thus, the descent condition for S_i can be obtained as [12]

$$\alpha_k \left(c_{S_i}^{k} \cdot d_{S_i}^{k}\right) \leq 0. \tag{15}$$

Since $\alpha_k > 0$, it can be rewritten as

$$\left(c_{S_i}^{k} \cdot d_{S_i}^{k}\right) \leq 0. \tag{16}$$

That is, (13) and (16) are equivalent.

From (7) and (8), it gets

$$c^k = \left[c_{S_1}^k, \ldots, c_{S_q}^k\right]^\top. \tag{17}$$

Since $d^k = [d_{S_1}^k, \ldots, d_{S_q}^k]^\top$, from (16) and (17) it gets that

$$c^k \cdot d^k = \sum_{i=1}^{q} \left(c_{S_i}^{k} \cdot d_{S_i}^{k}\right) \leq 0. \tag{18}$$

That is, the descent condition is also satisfied in design space S and it can be concluded that

$$f\left(x^{k+1}\right) \leq f\left(x^k\right). \tag{19}$$

Therefore, $\{f(x^k)\}$, is a nonincreasing sequence. Since $f(x^k)$ is a lower bounded function, it gets from (19) that

$$\begin{aligned} 0 &= \lim_{k\to\infty} \left[f\left(x^k\right) - f\left(x^{k+1}\right)\right] \\ &= \lim_{k\to\infty} \sum_{i=1}^{q} \left[f\left(x_{S_i}^{k}\right) - f\left(x_{S_i}^{k+1}\right)\right] \\ &\geq \lim_{k\to\infty} \sum_{i=1}^{q} \sigma_2\left(-c_{S_i}^{k} \cdot d_{S_i}^{k}\right) \geq 0, \quad \text{(by (12))}. \end{aligned} \tag{20}$$

Thus, from (20) and the forcing function definition, it gets

$$\lim_{k\to\infty} \left(-c_{S_i}^{k} \cdot d_{S_i}^{k}\right) = 0, \qquad \forall i \in (1, q). \tag{21}$$

Using (11) and (21), it gets

$$0 = \lim_{k\to\infty} \left(-c_{S_i}^{k} \cdot d_{S_i}^{k}\right) \geq \lim_{k\to\infty} \sigma_1\left(\left\|c_{S_i}^{k}\right\|\right) \geq 0. \tag{22}$$

Therefore, forcing function definition yield $\|c_{S_i}^k\| = 0$. From (17), it can be concluded that $\|c^k\| = 0$. That is, $\|\nabla f(x^k)\| = 0$, and the minimization or stationary solution can be obtained as

$$\lim_{k\to\infty} \left\|x^{k+1} - x^k\right\| = 0. \qquad \blacksquare$$

In general, finding the exact zero point of $\|\nabla f(x^k)\|$ is either impossible or impractical for numerical methods. Thus, the following convergence criteria can be applied to solve all the decomposed-space subproblems [13]:

I. Max $\{\|\nabla f_{S_i}(x_{S_i}^k, x_{\bar{S}_i})\|, i = 1, \ldots, q\} \leq \varepsilon$,
II. $\|\nabla f(x^k)\| \leq \varepsilon$, where $x^k = [x_{S_1}^k, \ldots, x_{S_q}^k]^\top$,
III. $|(x_{S_i}^{k+1})_j - (x_{S_i}^k)_j| \leq \varepsilon(|(x_{S_i}^{k+1})_j| + 1)$, where $(\cdot)_j$ denotes the j^{th} component of x_{S_i}.

It has been shown that Theorem 2.6 can be applied to minimization problem (1) even though the decomposed-space minimization functions are coupled to each other. Theorem 2.6 can be summarized as the following algorithm.

ALGORITHM 2.7. SPACE-DECOMPOSITION MINIMIZATION (SDM) ALGORITHM.

Step 1. Decompose the design space into a nonoverlapping decomposed-space set $\{S_1, \ldots, S_q\}$.

Step 2. Derive the decomposed-space minimization functions $f_{S_i}(x_{S_i}, x_{\bar{S}_i})$, for $i = 1, \ldots, q$.

Step 3. Choose the starting point x^1, where $x^1 = [x^1_{S_1}, \ldots, x^1_{S_q}]^\top$ and set $k = 1$.

Step 4. For $i = 1$ to q, solve one or more steps of the minimization subproblems $f_{S_i}(x^{k,i}_{S_i}, x_{\bar{S}_i})$ using any convergent algorithm, such as the conjugate gradient method, that satisfies descent conditions (11) and (12). Then, update $x^{k,i}_{S_i}$, which is the subvector of x^k; that is, $x^k = [x^{k,1}_{S_i}, \ldots, x^{k,q}_{S_q}]^\top$.

Step 5. Apply convergence criterion to all decomposed-spaces S_i. If the convergence criterion is satisfied for all decomposed spaces, then the minimization solution has been found as $x^* = [x^*_{S_1}, \ldots, x^*_{S_q}]^\top$; otherwise, set $k = k + 1$ and go to Step 4.

3. DIRECT-SEARCH SPACE-DECOMPOSITION MINIMIZATION (SDM) ALGORITHM

Since direct-search methods with no analytic gradient information are of interest to a number of researchers, a direct-search SDM algorithm is also introduced in this paper.

It has been shown that, if the design space S is further decomposed into a one-dimension decomposed-space set $\{S_1, \ldots, S_n\}$, $\forall S_i \in \Re^1$, one-dimensional search methods can be directly applied to Step 4 of Algorithm 2.7, and search direction updating can be eliminated. That is, the minimization function gradient is not required. The direct search algorithm is illustrated as the direct-search SDM algorithm below.

ALGORITHM 3.1. DIRECT-SEARCH SDM ALGORITHM.

Step 1 to 3 are the same as those in Algorithm 2.7 except for setting $q = n$.

Step 4. For $i = 1$ to n, solve the minimization solutions $x^{k,i}_{S_i}$ using any one-dimensional search method that satisfies

$$f_{S_i}\left(x^{k,i}_{S_i}, x_{\bar{S}_i}\right) \leq f_{S_i}\left(x^k_{S_i}, x_{\bar{S}_i}\right). \tag{23}$$

Then, update $x^{k,i}_{S_i}$ which is the subcomponent of x^k; that is, $x^k = [x^{k,1}_{S_1}, \ldots, x^{k,n}_{S_n}]^\top$.

Step 5. Check the convergence criteria

$$\max_{i \in (1,n)} \left| f_{S_i}\left(x^k_{S_i}, x_{\bar{S}_i}\right) - f_{S_i}\left(x^k_{S_i} + \delta, x_{\bar{S}_i}\right) \right| < \delta\varepsilon, \tag{24}$$

or

$$\sum_{i=1}^{n} \left[f_{S_i}\left(x^k_{S_i}, x_{\bar{S}_i}\right) - f_{S_i}\left(x^k_{S_i} + \delta, x_{\bar{S}_i}\right) \right]^2 < \delta^2\varepsilon^2, \tag{25}$$

where δ and ε are small positive values. If the convergence criterion has been satisfied, then the minimization solution has been found as $x^* = [x^*_{S_1}, \ldots, x^*_{S_n}]^\top$; otherwise set $k = k + 1$ and go to Step 4.

In the direct-search SDM algorithm, $x^k_{S_i}$ is used as the line search parameter, and any exact or inexact line search method satisfied (23) can be used. This is a special case of Algorithm 2.7. The advantages are that the search direction is either $+1$ or -1, depending only on the sign of $f_{S_i}(x^k_{S_i}, x_{\bar{S}_i}) - f_{S_i}(x^k_{S_i} + \delta, x_{\bar{S}_i})$, where δ is a small positive real number. Only one-dimensional search methods are used in every decomposed-space S_i, and the convergence of Algorithm 3.1 can

be proof by Theorem 2.6 with all the dimensions of decomposed space are set to one. In addition, if the gradient of any decomposed-space minimization function $\frac{df_{S_i}}{dx_{S_i}} = 0$ can be explicitly expressed as $x_{S_i} = g(x_{\bar{S}_i})$, then x_{S_i} can be calculated directly without using any one-dimensional search method.

4. HYBRID SPACE-DECOMPOSITION MINIMIZATION ALGORITHM

As shown in Figures 1 and 2, the direct-search SDM algorithm initially decreases the minimization functionís value far faster than the conjugate gradient method does. However, the direct-search SDM algorithm converges more slowly than the conjugate gradient method around

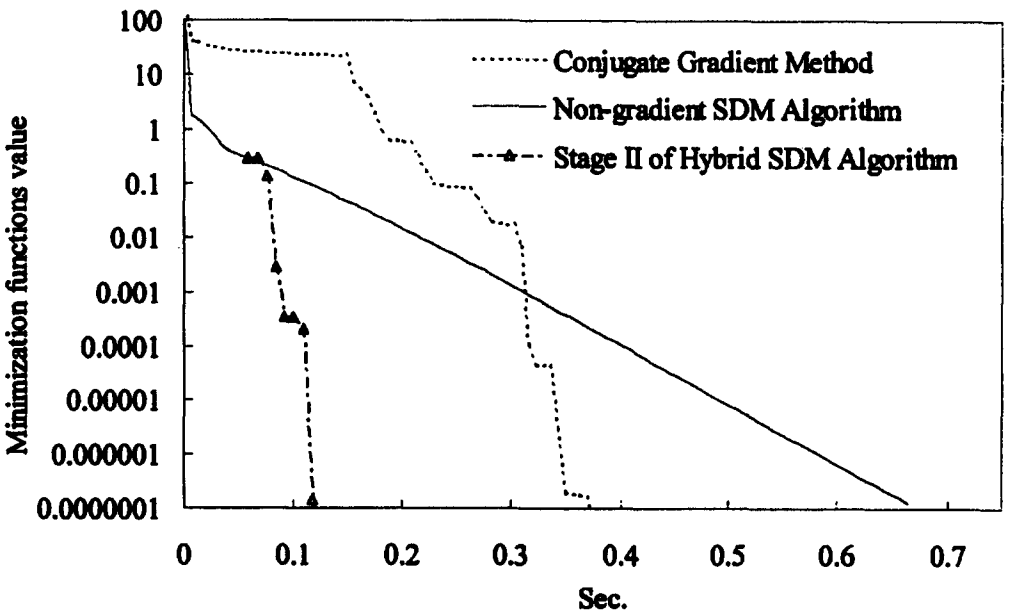

Figure 1. Minimization function value versus time for problem (6) when $n = 20$.

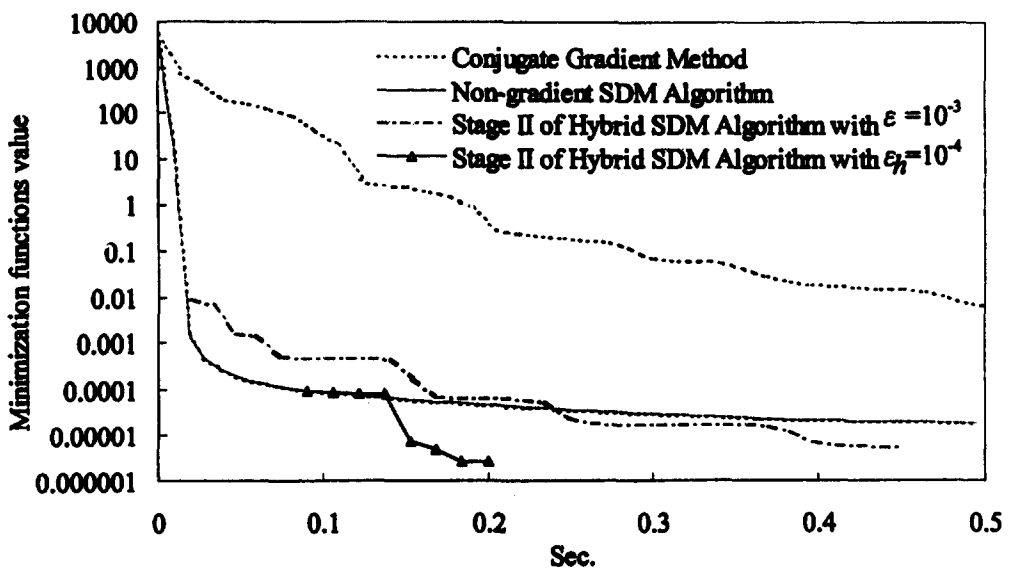

Figure 2. Minimization function value versus time for time for text problem (1) when $n = 100$.

the minimization point. Thus, a hybrid algorithm that combines the benefits of both algorithms is presented below.

ALGORITHM 4.1. HYBRID SPACE-DECOMPOSITION MINIMIZATION (HSDM) ALGORITHM.

Stage I

STEP 1 to 4 are the same as the direct-search Algorithm 3.1.

Step 5. If $f_{S_i}(x^k_{S_i}, x_{\bar{S}_i}) > 0$, $\forall k$, check the approximate convergence criterion

$$\frac{f_{S_i}\left(x^k_{S_i}, x_{\bar{S}_i}\right)}{f_{S_i}\left(x^{k+1}_{S_i}, x_{\bar{S}_i}\right)} < \varepsilon_h \frac{f_{S_i}\left(x^1_{S_i}, x_{\bar{S}_i}\right)}{f_{S_i}\left(x^2_{S_i}, x_{\bar{S}_i}\right)}, \tag{26}$$

otherwise, check the approximate convergence criterion

$$\max_{i\in(1,n)} \left| f_{S_i}\left(x^k_{S_i}, x_{\bar{S}_i}\right) - f_{S_i}\left(x^k_{S_i} + \delta, x_{\bar{S}_i}\right)\right| < \delta\varepsilon_h, \tag{27}$$

to all decomposed-spaces, where δ and ε_h are small positive values. If the approximate convergence criterion has been satisfied for all decomposed spaces, then go to Step 6; otherwise go to Step 4.

Stage II

Step 6. Using the approximate minimization solution in Stage I as the starting point, and then applying other convergent algorithms, such as the conjugate gradient method or the Algorithm 2.7, to find the final minimization solution.

Hybrid Algorithm 4.1 includes two stages. In the first stage, direct-search Algorithm 3.1 is used to solve for the approximate minimization solution. Then, another algorithm that converges more rapidly around the minimization point can be applied to solve for the final minimization solution in the second stage.

5. APPLICATION EXAMPLE

The SDM algorithms were applied to the learning of the single-layer perceptron neural network to demonstrate their effectiveness.

EXAMPLE 5.1. SINGLE-LAYER PERCEPTRON NEURAL NETWORK. The single-layer perceptron neural network with m input nodes and n output nodes shown in Figure 3 can be used as a pattern-classification device. The p^{th} input pattern $\{x_{0p}, \ldots, x_{mp}\}$ is multiplied by $\{w_{ji}\}$, which is a set of adjustable synaptic weights connecting the i^{th} input node to the j^{th} output node.

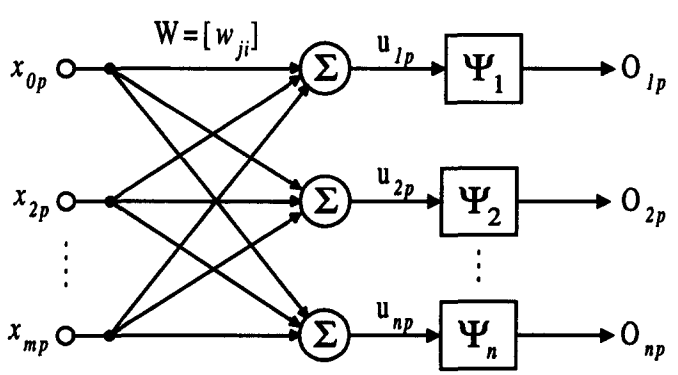

Figure 3. The single-layer perceptron neutral network.

Then, the internal potential u_{jp} generated at the j^{th} output node for the p^{th} input pattern, is evaluated using

$$u_{jp} = \sum_{i=0}^{m} (w_{ji} x_{ip}), \tag{28}$$

and the output $y_{jp} \in (1, -1)$ is generated either via hard limiter

$$y_{jp} = \text{sign } (u_{jp}),$$

or sigmoid function

$$y_{jp} = \tanh(\gamma u_{jp}), \qquad \gamma > 0. \tag{30}$$

Then, the output y_{jp} is compared with the desired signal, $d_{jp} \in (1, -1)$, and the quantized error

$$e_{jp} = d_{jp} - y_{jp} \tag{31}$$

is generated to adjust the weight.

The standard perceptron neural network learning algorithm minimizes the mean-squared error function. Such a mean-squared error function for the p^{th} pattern can be formulated as

$$E_p = \frac{1}{2} \sum_{j=1}^{n} e_{jp}^2, \tag{32}$$

and the total error function for all $\tilde{p}$ patterns is formulated as

$$E = \sum_{p=1}^{\tilde{p}} E_p = \frac{1}{2} \sum_{p=1}^{\tilde{p}} \sum_{j=1}^{n} e_{jp}^2. \tag{33}$$

In order to apply uncoupled space-decomposition Theorem 2.4 to minimize the total error function, (33) can be rewritten as

$$E = \frac{1}{2} \sum_{j=1}^{n} \sum_{p=1}^{\tilde{p}} e_{jp}^2 = \frac{1}{2} \sum_{j=1}^{n} E_j, \tag{34}$$

where $E_j = \sum_{p=1}^{\tilde{p}} e_{jp}^2 = \sum_{p=1}^{\tilde{p}} (d_{jp} - y_{jp})^2$ is the error function generated at the j^{th} output node for all input patterns.

Since y_{jp} is a function of $\{w_{j0}, \ldots, w_{jm}\}$ only, it can be concluded on the basis of Theorem 2.4 that the original design space S_w spanned by $\{w_{ji} \mid w_{ji} \in \Re^{(m+1)\times n},\ j = 1, \ldots, n,\ i = 0, \ldots, m\}$ can be decomposed into n nonoverlapping decomposed-spaces S_{w_j} spanned by

$$\{w_{ji} \mid w_{ji} \in \Re^m,\ i = 0, \ldots, m,\ \forall j \in (1, n)\}. \tag{35}$$

That is, $\bigcup_{j=1}^{n} S_{w_j} = S_w$ and $S_{w_j} \cap S_{w_{\tilde{j}}} = \emptyset$, if $j \neq \tilde{j}$. Then, the unconstrained minimization problem (34) can be decomposed into n uncoupled subproblems

$$E_j = \sum_{p=1}^{\tilde{p}} e_{jp}^2, \qquad \forall j \in (1, n). \tag{36}$$

These n uncoupled subproblems can be solved independently either on a single processor or on parallel processors. In addition, any of the n uncoupled subneural networks can be trained using either the conventional steepest-descent gradient rule [14] or some other more efficient minimization algorithm, such as the conjugate gradient method [15,16].

Furthermore, the direct-search Algorithm 3.1 or the hybrid Algorithm 4.1 can also be applied to these n uncoupled subproblems. To apply the direct-search Algorithm 3.1 or the first stage of Algorithm 4.1 to subproblems (36), the original design space S_{w_j} in (36) must be further decomposed into one-dimensional space $S_{w_{j\tilde{i}}}$ that is spanned by the subset

$$\left\{ w_{j\tilde{i}} \mid w_{j\tilde{i}} \in \Re^1,\ \forall \tilde{i} \in (1,n),\ \forall j \in (1,m) \right\}. \tag{37}$$

Then, the error function e_{jp} in (36) can be further decomposed into

$$\begin{aligned} e_{jp} &= d_{jp} - y_{jp} \\ &= d_{jp} - \tanh(\gamma u_{jp}) \\ &= d_{jp} - \tanh\left(\gamma \sum_{i=0}^{m} (w_{ji} x_{ip})\right) \\ &= d_{jp} - \tanh\left[\gamma\left(w_{j\tilde{i}} x_{\tilde{i}p} + \sum_{i\in(1,m) \text{ and } i\neq\tilde{i}} (w_{ji} x_{ip})\right)\right], \end{aligned}$$

where $\sum_{i\in(1,m) \text{ and } i\neq\tilde{i}}(w_{ji}x_{ip})$ is a constant value in the decomposed-space $S_{w_{j\tilde{i}}}$ and can be evaluated only once during the minimization process in decomposed-space $S_{w_{j\tilde{i}}}$.

6. NUMERICAL TESTING

To demonstrate the SDM algorithms, five large-scale test problems [10,11,17,18] were solved using the direct-search Algorithm 3.1 and the hybrid Algorithm 4.1. The numerical test results are shown in Tables 1 and 2, along with the results from the conjugate gradient method for comparison.

The notation used in Table 1 is shown below:

n = number of variables,
IT = number of iterations,
CPU = processor time measured in seconds.

The speed-up factors used in Table 2 were calculated as follows:

$$\text{Speed-up factor 1} = \frac{\text{Processor time for the conjugate gradient method}}{\text{Processor time for nongradient Algorithm 3.1}}, \tag{38}$$

$$\text{Speed-up factor 2} = \frac{\text{Processor time for the conjugate gradient method}}{\text{Processor time for hybrid Algorithm 4.1}}. \tag{39}$$

As shown in Table 2, the speed-up factors varied from 0.109 to 184.345 for the direct-search SDM algorithm, and from 1.5 to 27.708 for the hybrid SDM algorithm. The numerical results were obtained on a Pentium 120 Mhz machine with 48 M bytes of RAM memory, and the convergence criteria were set as $\varepsilon = 10^{-3}$ and $\varepsilon_h = 10^{-3}$. The numerical test results show that the conjugate gradient method may be superior to direct-search SDM algorithm on some test problems. As shown in Figures 1 and 2, the direct-search SDM has a high convergence characteristic during the first few steps, but the convergence speed slows down significantly near the minimization point. However, as shown in Figures 1 and 2, the slow convergence problem of the direct-search SDM can be improved using the hybrid SDM algorithm. As Figure 2 shows, if the switch point between the two stages of hybrid SDM Algorithm 4.1 is properly chosen, the processor time can be significantly reduced.

The processor times for the conjugate gradient method, the direct-search SDM algorithm and the hybrid SDM algorithm are shown in Figures 4–6, respectively. As Figure 1 shows, the

Table 1. CPU times and iteration numbers for the five test problems.

Problem	n	Direct-search SDM Algorithm		Conjugate Gradient Method*		Hybrid SDM algorithm*		
		IT	CPU	IT	CPU	IT1#	IT2##	CPU
1	20	90	0.041	67	0.480	7	20	0.130
	40	134	0.130	85	0.772	7	20	0.171
	100	214	0.500	110	1.502	7	32	0.450
	200	294	1.392	203	4.126	7	45	0.982
	400	394	3.646	371	12.828	7	46	1.742
	600	465	6.459	425	21.331	7	46	2.534
	800	521	9.654	456	29.292	7	47	3.315
	1000	569	13.199	459	35.651	7	47	4.056
2	20	13	0.005	66	0.571	3	31	0.290
	40	13	0.010	108	1.061	3	49	0.550
	100	14	0.040	210	2.934	3	51	0.801
	200	15	0.070	212	4.326	3	63	1.532
	400	15	0.140	416	13.670	3	68	2.794
	600	15	0.210	616	28.781	3	69	4.196
	800	16	0.291	814	46.547	3	69	5.307
	1000	16	0.380	1008	70.051	3	80	7.460
3	20	27	0.010	26	0.201	14	16	0.130
	40	32	0.030	42	0.330	14	26	0.220
	100	67	0.221	74	0.811	26	27	0.431
	200	95	0.581	113	2.063	38	15	0.631
	400	131	1.472	190	4.937	54	16	1.562
	600	143	2.093	341	14.340	57	17	2.804
	800	168	3.646	482	25.797	75	20	5.157
	1000	187	4.566	495	32.597	77	19	6.991
4	20	593	0.551	63	0.471	20	30	0.230
	40	615	1.172	73	0.620	20	42	0.391
	100	643	3.045	147	2.073	20	42	0.671
	200	664	6.229	246	5.838	20	44	1.092
	400	685	12.849	760	29.803	20	49	2.093
	600	697	19.608	1038	60.777	20	49	3.144
	800	706	26.468	1161	88.487	20	49	4.026
	1000	713	33.399	1447	134.303	20	48	4.847
5	20	959	0.631	53	0.371	6	19	0.140
	40	998	1.322	129	1.001	6	19	0.170
	100	1049	3.445	60	0.701	6	19	0.250
	200	1087	7.090	61	1.102	6	19	0.431
	400	1125	14.701	61	1.833	6	19	0.691
	600	1148	22.532	61	2.734	6	19	1.062
	800	1164	30.474	61	3.455	6	19	1.362
	1000	1176	38.475	61	4.186	6	20	1.702

*Gradient information is assumed to be available for the conjugate gradient method and hybrid SDM algorithm.
IT1 is the iteration number for Stage I of hybrid DSM Algorithm 4.1.
IT2 is the iteration number for Stage II of hybrid DSM Algorithm 4.1.

Table 2. Speed-up factors for the five test problems.

Problem	n	Speed-up Factor 1*	Speed-up Factor 2**
1	20	11.707	3.692
	40	5.938	4.515
	100	3.004	3.338
	200	2.964	4.202
	400	3.518	7.364
	600	3.303	8.418
	800	3.034	8.836
	1000	2.701	8.790
2	20	114.200	1.969
	40	106.100	1.929
	100	73.350	3.663
	200	61.800	2.824
	400	97.643	4.893
	600	137.052	6.859
	800	159.955	8.771
	1000	184.345	9.390
3	20	20.100	1.546
	40	11.000	1.500
	100	3.670	1.882
	200	3.551	3.269
	400	3.354	3.161
	600	6.851	5.114
	800	7.075	5.002
	1000	7.139	4.663
4	20	0.855	2.048
	40	0.529	1.586
	100	0.681	3.089
	200	0.937	5.346
	400	2.319	14.239
	600	3.100	19.331
	800	3.343	21.979
	1000	4.021	27.708
5	20	0.588	2.650
	40	0.757	5.888
	100	0.203	2.804
	200	0.155	2.557
	400	0.125	2.653
	600	0.121	2.574
	800	0.113	2.537
	1000	0.109	2.459

*Speed-up factor 1 was calculated using (41).

**Speed-up factor 2 was calculated using (42).

processor times for the conjugate gradient method are approximately exponential functions of the design variable numbers. By contrast, the processor times for the direct-search and hybrid SDM algorithms are approximately linear functions of the design variable numbers, as shown in Figures 5 and 6.

In order to compare the memory resources required by these methods, a Memory-Required Ratio (MRR) were calculated as follows:

$$MRR = \frac{\text{Memory required for nongradient DSM Algorithm 3.1}}{\text{Memory required for the conjugate gradient method}}. \tag{40}$$

For minimization problem (1) with n design variables, it can be shown that the conjugate gradient method requires at least $3n$ units of memory to save the design variables, minimization function gradient and search-direction vector. Furthermore, additional temporary memory is required during the computation process for the conjugate gradient method. By contrast, the

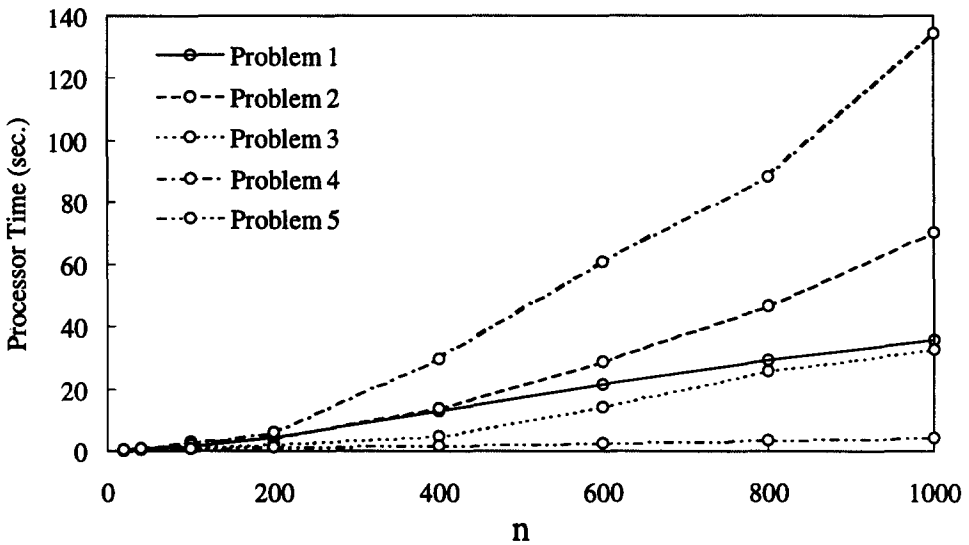

Figure 4. Processor times for the conjugate gradient method for different scales.

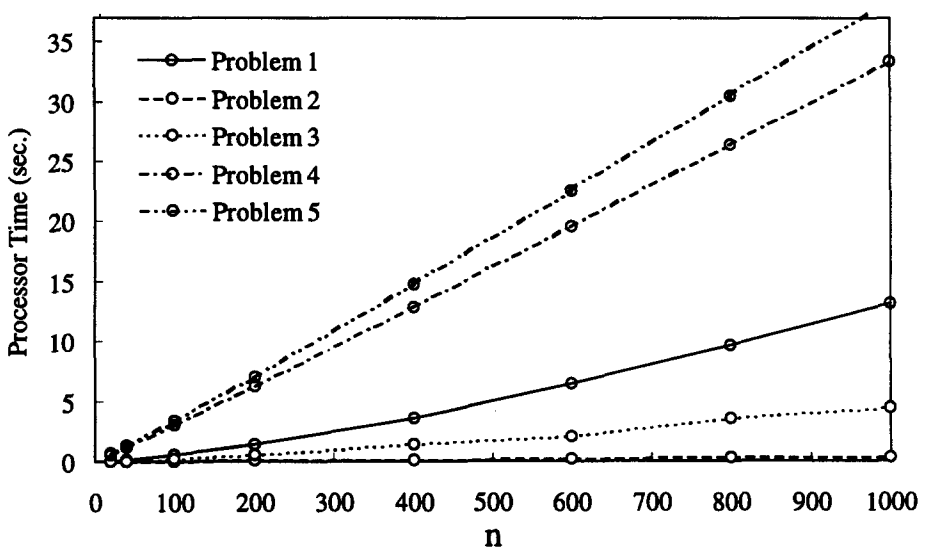

Figure 5. Processor time for the nongradient DSM algorithm for different scales.

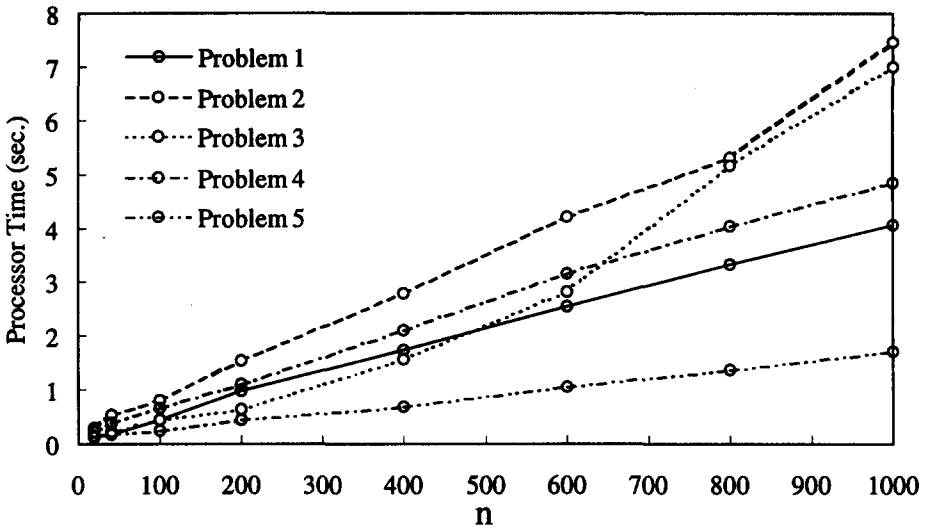

Figure 6. Processor time for the hybrid DSM algorithm for different scales.

direct-search SDM algorithm requires only n units of storage memory to save the design variables and some more units of temporary storage for the one-dimensional search method.

Because temporary storage requirements are strongly dependent on programming techniques, the minimum amount of memory required was used to calculate the MRR. The MRR is less than 0.3333 as compared with conjugate gradient method. Therefore, the direct-search SDM algorithm is particular suitable for large-scale problems due to its low memory requirement.

7. CONCLUSIONS

Three fundamental convergent space-decomposition minimization (SDM) algorithms for large-scale unconstrained minimization problems have been presented in this paper. These algorithms allow minimization problems to be decomposed into q subproblems. If the decomposed-space minimization functions are uncoupled from each other, the q subproblems can be solved independently using any convergent algorithm. However, if they are coupled to each other, the iterative space-decomposition minimization (SDM) algorithm can be used and all subproblems can be solved iteratively with any convergent algorithm that satisfies space-decomposition Theorem 2.6. Furthermore, it has been shown that if the design space is further divided into one-dimensional decomposed spaces, general one-dimensional search methods can be applied directly to the one-dimensional subproblems, and the SDM algorithm can be considered a direct-search algorithm. Although the direct-search SDM algorithm converges slowly near the minimization point, the hybrid SDM algorithm can be used to eliminate the slow convergence problem.

Numerical tests have shown that the SDM algorithms save more processor time and memory resources than the conjugate gradient method. In addition, properly choosing the switch point between the two stages of the hybrid SDM algorithm allows the processor time to be significantly reduced, and further study of methods for finding the proper switch point for the hybrid SDM algorithm is warranted.

Although all of the test problems are solved on a serial computer, all of the algorithms presented in this paper are particularly suitable for parallel computers after further modification. Further study and testing on parallel computer of the SDM algorithms are warranted.

In the application example, the single-layer perceptron neural network was used as an example to demonstrate the effectiveness of SDM algorithms. However, the SDM algorithms can be extended to multilayer perceptron neural network by the multilevel decomposition methods [19,20] and further study and testing for multilayer perceptron neural network are also warranted.

APPENDIX

PROBLEM 1. EXTENDED POWELL TEST FUNCTION [9,10].

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

$$x^{(1)} = [3, -1, 0, 1, \ldots, 3, -1, 0, 1,]^\top.$$

PROBLEM 2. EXTENDED DIXON TEST FUNCTION [11].

$$F = \sum_{i=1}^{n/10} \left[(1 - x_{10i-9})^2 + (1 - x_{10i})^2 + \sum_{j=10i-9}^{10i-1} \left(x_j^2 - x_{j+1} \right)^2 \right],$$

$$x^{(1)} = [-2, -2, \ldots, -2]^\top.$$

PROBLEM 3. TRIDIA TEST FUNCTION [10].

$$F = \sum_{i=2}^{n} [i(2x_i - x_{i-1})^2],$$

$$x^{(1)} = [3, -1, 0, 1, \ldots, 3, -1, 0, 1]^\top.$$

PROBLEM 4. EXTENDED WOOD TEST FUNCTION [9,11].

$$F = \sum_{i=1}^{n/4} \Big\{ 100 \left(x_{4i-3}^2 - x_{4i-2} \right)^2 + (x_{4i-3} - 1)^2 + 90 \left(x_{4i-1}^2 - x_{4i} \right)^2 + (1 - x_{4i-1})^2 + 10.1 \left[(x_{4i-2} - 1)^2 + (x_{4i} - 1)^2 \right] + 19.8(x_{4i-2} - 1)(x_{4i} - 1) \Big\},$$

$$x^{(1)} = [-3, -1, -3, -1, \ldots, -3, -1, -3, -1]^\top.$$

PROBLEM 5. EXTENDED ROSENBROCK TEST FUNCTION [10,11].

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

$$x^{(1)} = [-1.2, 1, -1.2, 1, \ldots, -1.2, 1]^\top.$$

REFERENCES

1. M.C. Ferris and O.L. Mangasarian, Parallel variable distribution, *SIAM J. Optim.* **4** (4), 815–832, (1994).
2. M.V. Solodov, New inexact parallel variable distribution algorithms, *Computational Optim. and Applications* **7**, 165–182, (1997).
3. O.L. Mangasarian, Parallel gradient distribution in unconstrained optimization, *SIAM J. Contr. & Optim.* **33** (6), 1916–1925, (1995).
4. V.M. Kibardin, Decomposition into functions in the minimization problem, *Automation and Remote Control* **40** (1), 1311–1323, (1980).
5. K. Mouallif, V.H. Nguyen and J.-J. Strodiot, A perturbed parallel decomposition method for a class of nonsmooth convex minimization problems, *SIAM J. Contr. & Optim.* **29** (4), 829–847, (1991).
6. D.G. Mcdowell, Generalized conjugate directions for unconstrained function minimization, *J. Optim. Theory and Applications* **41** (4), 523–531, (1983).
7. Y.A. Shpalenskii, Iterative aggregation algorithm for unconstrained optimization problems, *Automation and Remote Control* **42** (1), 76–82, (1981).
8. C. Sutti, Nongradient minimization methods for parallel processing computers, Parts 1 and 2, *J. Optim. Theory and Applications* **39** (4), 465–488, (1983).

9. E.C. Housos and O. Wing, Pseudo-conjugate directions for the solution of the nonlinear unconstrained optimization problem on a parallel computer, *J. Optim. Theory and Applications* **42** (2), 169–180, (1984).
10. A. Buckley and A. Lenir, QN-link variable storage conjugate gradients, *Math. Programming* **27**, 155–175, (1983).
11. D. Touati-Ahmed and C. Storey, Efficient hybrid conjugate gradient techniques, *J. Optim. Theory and Applications* **64**, 379–397, (1990).
12. J.S. Arora, *Introduction to Optimum Design*, McGraw-Hill, New York, (1989).
13. P.J.M. van Laarhoven, Parallel variable metric algorithms for unconstrained optimization, *Math. Programming* **33**, 68–81, (1985).
14. A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, John Wiley & Sons, New York, (1993).
15. C. Charalambous, Conjugate gradient algorithm for efficient training of artificial neural networks, *IEE Proceedings Part G* **139** (3), 301–310, (1992).
16. E.M. Johansson, F.U. Dowla and D.M. Goodman, Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method, *International J. of Neural Systems* **2** (4), 291–301, (1992).
17. J.J. Moré, B.S. Garbow and K.E. Hillstrom, Testing unconstrained optimization software, *ACM Trans. Math. Software* **7** (1), 17–41, (1981).
18. J.A. Snyman, A convergent dynamic method for large minimization problems, *Computers Math. Applic.* **17** (10), 1369–1377, (1989).
19. W. Xicheng, D. Kennedy and F.W. Williams, A two-level decomposition method for shape optimization of structures, *Int. J. Numerical Methods in Engineering* **40**, 75–88, (1997).
20. U. Kirsch, Two-level optimization of prestressed structures, *Engineering Structures* **19** (4), 309–317, (1997).