------ Raf. Jour. Sci., Vol.17, No.1, pp.50- 58, 2006 ------

A Parallel Optimization Method Based on the Theory of Parallel Subspaces

Bashir M. Khalaf
Department of Mathematics
College of Education
Mosul University

Khalil K. Abbo
College of Computer Science
& Mathematics
Mosul University

(Received 12/8/2001 , Accepted 27/10/2001)

ABSTRACT

The purpose of this paper is to develop a parallel algorithm for solving unconstrained optimization problems. This parallel algorithm in which several tasks may be executed at the same time in parallel based on the parallel subspaces theorem and it's designed to run on MIMD computing systems.

طريقة أمثلية متوازية مبنية على نظرية الفضاءات الجزئية المتوازية

الملخص

تم في هذا البحث تطوير خوارزمية متوازية في الأمثلية غير المقيدة أعتمد على نظرية الفضاءات الجزئية المتوازية. هذه الخوارزمية مصممة للحاسبات المتوازية من نوع MIMD .

INTRODUCTION

In recent years as microprocessor have become cheaper and technology for interconnecting them has improved, it has become possible and practical to build general purpose parallel computers containing a large number of processors. There has been burts of activity in the developing the hardware, the algorithms and theoretical models to make use of parallel computers.

In this paper we discuss the development of parallel algorithm based on steepest descent and parallel subspace algorithms, designed to run on MIMD (Multiple Instruction Multiple Data) system. The MIMD system are consisting of several processors where each processor can independently run it's own instructions and these processors are connected with each other by suitable communication network, for more detail see (Khalaf and Hutchison, 1991) and (Khalaf and Hutchison, 1992).

For minimizing differentiable non-linear functions consider the unconstrained optimization problem:

$$\text{Minimize } f(x), \qquad x \in R^n \qquad (1)$$

Where $f(x)$ is objective function, assumed to be continuously differentiable and denote to the gradient vector $\nabla f(x)$ by $g(x).$ one of the oldest and most widely known methods for solving eq(1) is the method of Steepest Descent (often refered to as the gradient method). The method is extremely important from theoretical view point. Since it's one of the simplest for which satisfactory analysis exists and it is behavior for general function is similar to it's behavior for quadratic function (Luenberger, 1973).

The summary of Steepest Descent is given below:

Algorithm (1) (Steepes Descent.algorithm).

It is assumed that an estimate x_0 of a minimizer x^* of f is known and set tolearns $\varepsilon > 0$

Step-1: set k = 0

Step-2: compute d_k from

$$d_k = -g_k \qquad (2)$$

Step-3: compute α_k from

$$f(x_k + \alpha_k d_{k)} = \min_\alpha f(x_k + \alpha\ d_{k)} \qquad (3)$$

Step-4: compute x_{k+1} from

$$x_{k+1} = x_k + \alpha_k d_k \qquad (4)$$

Step-5: if $\|g_{k+1}\| \le \varepsilon$, stop otherwise

Set k = k+ l and go to step 2

Search direction $d_1, d_2, ..., d_k$ generated by algorithm (1) are downhill also the sequence $\{x_k\}$ generated by Steepest descent algorithm. Converge to point x at which $g(x).$=0. See (Wolfe, 1978).

Theorem (1) (Steepest Descent-Quadratic Case)

For any $x_0 \in \mathrm{R}^n$ algorithm (1) converges to the unique minimum point x^* of f. furthermore with

$$E(x) = \tfrac{1}{2}(x - x^*)^T Q(x - x^*)$$

there holds at every step k

$$E(x_{k+1}) \le \left(\frac{A-a}{A+a}\right)^2 E(x_k)$$

Where a and A are respectively the smallest and largest eigenvalues of nxn positive definite matrix Q.

(For proof see Luenberger, 1973). In general the convergent property which is derived for quadratic problem in theorem (1) can be translated into similar one for non-quadratic problem. (Luenberger, 1973).

1. Parallel subspaces Method:

The parallel subspaces algorithms depend upon the parallel subspaces theorem. The first algorithm which used this theorem was proposed by Smith (1962), later in 1964 powell proposed another algorithm based on parallel subspaces theorem.

Hestenes 1980 re-defined the subspaces theorem in terms of parallel planes, which states as follows.

Theorem (2):

Let x_k and $\bar{x}_k$ be the minimum points of F where F, is a quadric function on two distinct parallel (k -1)-planes $\prod_{k-1}$ and $\overline{\Pi_{k-1}}$. The vector

$$d_k = x_k - \bar{x}_k \qquad (5)$$

Is conjugate to these (k-1) planes. The minimum point x_{k+1} of F on the line $x = x_k + \alpha d_k$ through x_k and $\bar{x}_k$ affords a minimum to F on the K-plane $\Pi k = \Pi k\text{-}1 + \alpha d_k$. spanning Π_{k-1} & $\overline{\Pi_{k-1}}$.

for the proof see Hestenes 1980.

This result is represented schematically in figure (1).

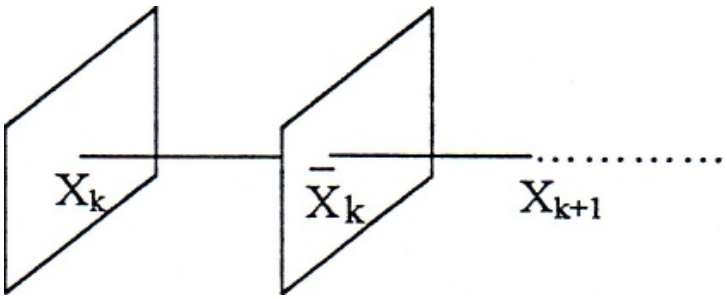

Figure 1

The result described in theorem (2) suggests that the minimum point of f can be found by the following procedure.

Select an initial point x_1 and obtain the minimum point x_2 of f on a line Π_1 through x_1. Next find the minimum point $\overline{X_2}$ of f on the line $\overline{\Pi_1}$ parallel and distinct from Π_1. Then minimize f on the line joining x_2 to $\overline{X_2}$ to obtain the minimum point x_3 of f on the 2-plane Π_2 spanning Π_1 and $\overline{\Pi_1}$. W e proceed by finding minimum point $\overline{X_3}$ on a 2-plane $\overline{\Pi_2}$ parallel to Π_2 and determining the minimum point x_4 on the line joining x_3 to $\overline{X_3}$.

The point x_4 minimizes f on 3-plane Π_3 spanning Π_2 and $\overline{\Pi_2}$. Proceeding in this manner we obtain the minimum points x_2, $x_3 \ldots x_{n+1}$ of f successively on planes $\Pi_1, \Pi_2 \ldots, \Pi_n$ since Π_n is the whole space the minimum point x_{n+1} of f on Π_n is the minimum point x^* of f. The procedure just described was the method of parallel subspaces.

2. New Parallel Optimization Method

Steepest Descent and Parallel Subspace methods mentioned earlier can be combined in a parallel optimization algorithm to run on machines that have more than one processors working on one problem at the same time to reduce the solution time of the problem by parallel processing.

We well use some simplifying notations such as X_i^k which means that the value of x in processor i at iteration k, similar notation used for gradient vector for g(X_i^k) = ∇f (X_i^k). We can now summarize the new algorithm as follows select initial points X_i^0, i=1,....., m where m is the number of the processors contained in the computer and select an arbitrary directions say d_i^0 where $d_1^0 // d_2^0 // \ldots // d_m^0$. To obtain the minimum points $X_i^{(1)}$(i = 1,2,.....,m) of the objective function on the parallel and distinct lines Π_i (i = 1..... m) see figure (2).

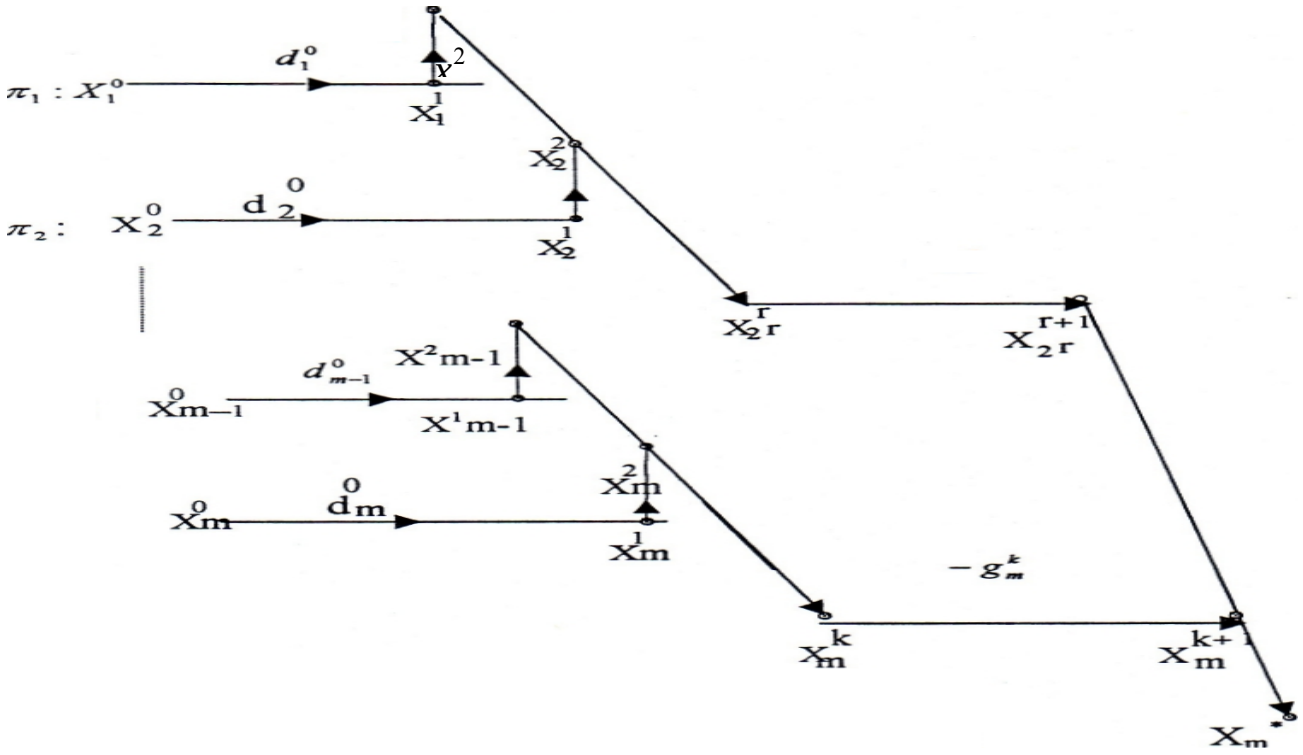

Fig.2: The diagram of the new method

Next fined new search directions from

$d_i^k = -g_i^k$ k > o ……..(6)

perform a line search along the directions d_i^k to find new points

$x_i^{k+1} = x_i^k + \alpha_i^k d_i^k$ …… (7)

where α_i^k optimal step size. For next iteration

Set $d_i^{k+1} = x_{2i}^{k+1} - x_{2i-1}^{k+1}$ i = 1,....., m/2 .....(8)

Then minimize f on the directions d_i^{k+1} to obtain the minimum points X_i^{k+2}. Proceeding in this manner we obtain the minimum point X_m^* of the objective function. The outline of the algorithm given below

Algorithm (2)

set: k = l, x_i^{k-1}, d_i^{k-1}, ε, (i = l, ..., m)

fined x_i^k (i = 1..... m) from equation (7)

Step-1: calculate g_i^k and fined d_i^k from equ. (6)

Step-2: use eq (7) to obtain new points

Step-3: if $\| g_i^{k+1} \| < \varepsilon$ for some i, stop. Otherwise continue

Step-4: use eq. (7) and (8) to find new d_i^k and X_i^k i = 1, . . ., m/2

Step-5: k = k + l, m = m/2 go to step-1

The steps (1) to (5) are repeated until the point X_m^t (t >k) is obtained in processor p_m. If X_m^t is not the minimum point then p_m well finds new direction say d_m^t and sends the value of d_m^t to processors p_1, ..., p_{m-1} as initial direction and processes repeated.

To run the algorithm (2) on parallel computer we must connect the processors as seen in figure (3)

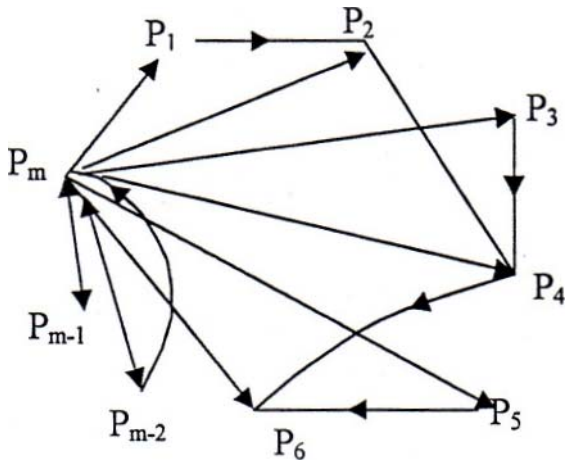

Fig. 3: The communication path of the processors

Where the processors P_1,…, P_m (for no lose of generality we assume m is even) are operating in parallel and computing in two stages :

First stage: all processors are runs and computes $x_i^{(1)}$, $g_i^{(1)}$, $d_i^{(1)}$ and $x_i^{(2)}$, i=1,…,m, if convergent is not obtained, then each processor p_{2i-1} sends values to processor P_{2i} (i = l, ... m/2) second stages processor P_{2i} receives values from p_{2i-1} i = 1,... ...., m/2 to compute new directions from eq (8) and new points from eq (7). Then check for convergent otherwise processor p_{2i-2} sends values to P_{2i}. The process repeated until X_m^{k+1} obtain in the P_m.

We can formalize this parallel algorithm on MIMD computing systems as follows:

P_1
1- set $X_1^0, d_1^0, \varepsilon$
2- for k≥1 find X_1^k, g_1^k, d_1^k
3- compute X_1^{k+1} from line search
4- save X_1^{k+1} as X_1^0 and send the value of X_1^{k+1} to p2
5- receive value from Pm
6- if convergent is not obtained ? go to 2

P_2
1- set $X_2^0, d_2^0, \varepsilon$
2- for k≥1 find X_2^k, g_0^k, d_2^k
3- compute X_2^{k+1} from line search
4- receive value of X_1^{k+1} by P1 and find new d_2^{k+1} from $d_0^{k+1} = X_2^{k+1} - X_1^{k+1}$
5- compute X_2^{k+2} by line search
6- save X_2^{k+2} as X_2^0 and send the value of X_0^{k+1} to P4
7- receive value from P_m
8- if convergent is not obtained go to 2

To p_4

P_{m-2}
1- set $X_{m-2}^0, d_{m-2}^0, \varepsilon$
2- for K≥1 find $X_{m-2}^k, g_{m-2}^k, d_{m-2}^k$
3- compute X_{m-2}^{k+1} from line search
4- receive value from P_{m-3} and find new d_{m-2}^{k+1} from $d_{m-2}^{k+1} = V_{m-2}^{k+1} - X_{m-2}^{k+1}$
5- compute X_{m-2}^{k+2} by line search
6- save X_{m-2}^{k+2} as X_{m-2}^0 and send the value of X_{m-2}^{k+2} to Pm
7- receive value from P_m
8- if convergent is not obtained go to 2

From P_{m-3}

P_{m-1}
1- set $X_{m-1}^0, d_{m-1}^0, \varepsilon$
2- for K≥1 find $X_{m-1}^k, g_{m-1}^k, d_{m-1}^k$
3- compute X_{m-1}^{k+1} by line search
4- save X_{m-1}^{k+1} as X_{m-1}^0 and send the value of X_{m-1}^{k+1} to Pm
5- receive value from P_m
6- if convergent is not obtained go to 2

P_m
1- set $X_m^0, d_m^0, \varepsilon$
2- for K≥1 find X_m^k, g_m^k, d_m^k
3- compute X_m^{k+1} from line search
4- receive value from P_{m-1} and find new d_m^{k+1} from $d_m^{k+1} = x_m^{k+1} - x_{m-1}^{k+1}$
5- compute X_m^{k+2} by line search
6- if convergent is not obtained find new d_m^{k+2} and send the value of d_m^{k+2} to processors p_1, ……, p_{m-1} as intial direction and go to 2

3- Numerical Examples:

Since the parallel computers are not available, we tried to solve two examples by hand, for simplicity we use two processors (m=2).

Let us consider the following unconstrained optimization problems (See Mokhtar, 1993).

Example (1):

Minimize $f(X_1\text{-}X_2)=(2X_1\text{-}X_2)^2+(X_2+1)^2$

Processor (1)	**Processor (2)**
1- set $x_1^0=\begin{pmatrix}2.5\\2\end{pmatrix}, d_1^0=\begin{pmatrix}1\\0\end{pmatrix}, \varepsilon$	1- set $x_2^0=\begin{pmatrix}1\\3\end{pmatrix}, d_2^0=\begin{pmatrix}1\\0\end{pmatrix}, \varepsilon$
2- fined x_1^1, g_1^1, d_1^1 $x_1^0=\begin{pmatrix}1\\2\end{pmatrix}, g_1^1=\begin{pmatrix}0\\5\end{pmatrix}, d_1^1=\begin{pmatrix}0\\5\end{pmatrix}$	2- fined x_2^1, g_2^1, d_2^1 $x_2^1=\begin{pmatrix}1.5\\3\end{pmatrix}, g_2^1=\begin{pmatrix}0\\7\end{pmatrix}, d_2^1=\begin{pmatrix}0\\-7\end{pmatrix}$
3- from d_1^1 fined x_1^2 $x_1^2=\begin{pmatrix}1.0\\0.5\end{pmatrix}$	3- from d_2^1 fined x_2^2 $x_2^2=\begin{pmatrix}1.5\\1\end{pmatrix}$
4- check for convergent	4- receive value from P_1
5- send value of x_1^2 to P_2	5- fined new search direction from $d_2^2=x_2^2-x_1^2=\begin{pmatrix}0.5\\0.5\end{pmatrix}$
	6- fined new x_2^3 from $X_2^3=X_2^2-\alpha_2^2=\begin{pmatrix}-0.5\\-1\end{pmatrix}$ which is true minimum for $f(X_1,X_2)$

Example (2):
Minimize $f(X_1-X_2)=(X_1-2)^4+(X_1-2X_2)^2$

Processor (1)	**Processor (2)**
1- set $x_1^0=\begin{pmatrix}0\\3\end{pmatrix}, d_1^0=\begin{pmatrix}1\\0\end{pmatrix}, \varepsilon$	1- set $x_2^0=\begin{pmatrix}-1\\-2\end{pmatrix}, d_2^0=\begin{pmatrix}1\\0\end{pmatrix}, \varepsilon$
2- calculate x_1^1 from $x_1^1=x_1^0+\alpha_1^0 d_1^0$ $\alpha_1^0=3.128, x_1^1=\begin{pmatrix}3.123\\3\end{pmatrix}$	2- calculate x_2^1 from $x_2^1=x_2^0+\alpha_2^0 d_2^0$ $\alpha_2^0=1.673, x_2^1=\begin{pmatrix}0.673\\-2\end{pmatrix}$
3- fined $g_1^1, d_1^1=-g_1^1, g_1^1=\begin{pmatrix}-0.003\\11.488\end{pmatrix}$ fined $x_1^2=x_1^1+\alpha_1^1 d_1^1, x_1^2=\begin{pmatrix}3.1276\\1.56437\end{pmatrix}$	3- fined $g_2^1, d_2^1=-g_2^1, g_2^1=\begin{pmatrix}-0.001\\-18.692\end{pmatrix}$ fined $x_2^2=x_2^1+\alpha_2^1 d_2^1, x_2^2=\begin{pmatrix}0.673\\-2\end{pmatrix}$
4- send value of x_1^2 to processor 2	4- receive value from P_1
	5- fined d_2^2 from $d_2^2=x_2^2-x_1^2$ $d_2^2=\begin{pmatrix}-2.4546\\-1.2278\end{pmatrix}$
	6- $X_2^3=X_1^2-\alpha_2^2 d_2^2=\begin{pmatrix}2\\1\end{pmatrix}$ which is true minimum for $f(X_1,X_2)$

From the solution of the examples (1) and (2) we see that in the first iteration two processors are runs and obtains two minimum points X_1^2 in p_1 and X_2^2 in p_2, then processor p_1 sends the value of X_1^2 to processor p_2 to compute new direction d_2^2 and new minimum point X_2^3 which is true minimum point. These examples shows the new method reduces the solution time required to solve the problem, clearly the total time needed to solve any problem depends on the number of the processors.

CONCLUSIONS

In this paper, we had developed a new method using the theory of steepest descent and theory of parallel subspace method. The parallel tasks are illustrated by mean of practical examples.

The expected speed – up factor of the new method is demonstrated.

REFERENCES

Hestenes, M.R., 1980. Conjugate Direction Method in Optimization, Spring-Verlage. New York, Heiddberg. Berlin.

Khalaf, B.M. and Hutchison, 1991. Parallel Algorithm for Solving IVPs, J. of Parallel Computing. Vol. 17. 957 p.

Khalaf, B.M. and Hutchison, 1992 Redusing the Solution Time of Liner Meth. Models by Using a Transputer Applications Based System. Parallel Computing and Transputer Applications M. Valero et al (Eds))Barcelona. pp148.

Luenberger, D.C. ,1973. Introduction to Linear and non – Linear Programming. Addison Wesley. Reading. Mass.

Mokhtare, B.S., 1993. Non-Linear Programming Theory and Application, by John Wiley and Sons. Inc. Second Edition.

Powell, M.J.D., 1964. An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives,Computer Journal. Vol. (6).

Smith, C., 1962. The Automatic Computation of Maximum Like-Lihood estimates, Wolfe, A.M., 1978. Numerical Methods for Unconstrained Optimization. An Introduction. By Van Nostrand Company. Englewood Cliffs. N.J. USA.