Parallel Direct Search Methods

Bashir M. Khalaf — *College of Education*
Mohammed W. Al-Neama — *Education College for Girls*
University of Mosul

Received on:19/8/2010 **Accepted on:10/11/2010**

ABSTRACT

Mostly minimization or maximization of a function is very expensive. Since function evaluation of the objective function requires a considerable time. Hence, our objective in this work is the development of parallel algorithms for minimization of objective functions evaluation takes long computing time. The base of the developed parallel algorithms is the evaluation of the objective function at various points in same time (i.e. simultaneously).

We consider in this work the parallelization of the direct search methods, as these methods are non-sensitive for noise and globally convergent. We have developed two algorithms mainly they are dependent on the Hock & Jeff method in unconstrtrained optimization.

The developed parallel algorithm are suitable for running on MIMD machine which are consisting of several processors operating independently, each processor has it's own memory and communicating with each other through a suitable network.

Key-Words: nonlinear optimization, unconstrained optimization, multidirectional search, parallel direct search, parallel computing, MIMD Computers.

الطرائق المتوازية للبحث المباشر

بشير خلف — **كلية التربية**
محمد النعمة — **كلية التربية للبنات**
جامعة الموصل

تاريخ الاستلام: 2010/8/19 **تاريخ القبول: 2010/11/10**

الملخص

إيجاد اقل أو اكبر قيمة لدالة يكون على الأغلب مكلفاً جدا، إذ أن حساب قيمة الدالة في نقطة ما يأخذ وقتاً طويلاً.هدفنا في هذا العمل هو تطوير خوارزميات متوازية لإيجاد اقل أو أكبر قيمة للدوال التي تحتاج حساب قيمها وقت طويل، وأساس هذا التطوير هو إيجاد قيمة دالة الهدف في نقاط مختلفة في آن واحد.

درسنا في هذا العمل تطوير خوارزميات متوازية لطرائق البحث المباشر كون هذه الطرائق غير حساسة لتشويش ومتقاربة بصورة عامة، قمنا بتطوير خوارزميتين مبنيتين بالأساس على طريقة هوك وجيف في الأمثلية غير المقيدة.

الخوارزميات المتوازية المطورة مناسبة للتنفيذ في حاسبات من نوع MIMD والتي تتكون من عدة معالجات مستقلة وكل معالج له ذاكرة خاصة له وتتصل المعالجات مع بعضها من خلال شبكة اتصال مناسبة.

الكلمات المفتاحية: الامثلية غير الخطية ، الامثلية غير المقيَّدة ، البحث متعدد الاتجاهات ، البحث المباشر الموازي ، الحوسبة المتوازية ، حواسيب MIMD.

1. Introduction:

Optimization is a mathematical discipline which appears in many fields such as engineering, economics, operations research, management science, etc. Such as maximizing the production of rice, reducing cost of a car, or getting best performance out of a battery. Optimization can be described as a method of getting best out of any situation. Formally, optimization is minimization or maximization of a function subject to certain constraints. Mathematically we represent an optimization problem as: maximize $f(x)$ subject to $x \in D$,
or minimize $f(x)$ subject to $x \in D$.

The function $f: R^n \rightarrow R$ is called the objective function, and set $D \subset R^n$ the constraint set. $x = [x_1, x_2, ..., x_n]^T$ is the vector representing n independent variables. Very often optimization problem is stated as minimization problem. An optimization "problem" is unconstrained if the constraints do not have any effect at optimum.(1)

Today optimization is well understood discipline with rigorous analysis methods. But in early 60's, the tools and techniques of analysis were yet to be developed and proven. In 1961, Robert Hooke and T.A. Jeeves developed a method for optimization and coined the phrase "direct search" [5], [13]. They provided the following description of direct search methods in the introduction of the paper:

We use the phrase "direct search" to describe sequential examination of trial solutions involving comparison of each trial solution with the "best" obtained up to that time together with a strategy for determining (as a function of earlier results) what the next trial solution will be. The phrase implies our preference, based on experience, for straightforward search strategies which employ no techniques of classical analysis except where there is a demonstrable advantage in doing so.

Hooke and Jeeves's paper appeared before any of the "techniques of classical analysis" that use Taylor series expansion of the objective function became available. Objective function can be expanded using Taylor series expansion as:

$$f(x + \Delta x) = f(x) + \nabla x^T \Delta x + \frac{1}{2} \Delta x^T H \Delta x + \ldots$$

where Δx is a vector of variable increments, ∇x is the gradient vector containing the first partial derivatives, and H is the matrix of second partial derivatives, the Hessian matrix.

Direct search methods neither require nor estimate derivatives. As a consequence, while they are usually slower to converge than derivative based methods, they are usually much more robust in situations where the function values are subject to noise, analytic derivatives are unavailable, or finite difference approximations to the gradient are unreliable. Furthermore, the direct search schemes given here parallelize very well although they can certainly be used as sequential methods.[7, 10]

Hence the objective of this research is the development of a parallel Direct search methods which is suitable for running on a MIMD (Multiple Instruction streams with Multiple Data streams) computer ([6,8,9,11]).

(1) Though the term unconstrained, is standard, is somewhat misleading and does not mean lack of constraints. It refers to a situation in which one can move a small distance away from the optimum point in any direction without leaving the feasible region [12].

MIMD Computer Consists of several processors, each processor has its own memory and processing unit. These processors communicate through a suitable communication network. (for more detail, see [1,2,8,11]). In MIMD computer each processor can carry out its own set of instructions, often on its own set of data, independently of all the other processors. Such computers usually number their (more complex) processors in tens rather than thousands that may be found in SIMD (Single Instruction Stream with Multiple Data Stream) computers. MIMD computers are well suited to algorithmic parallelism in which problems can be separated into concurrent independent processors [4].

2. Direct Search methods:

Direct search method as described by Hooke and Jeeves [13] requires space of points $P \in R^n$ (henceforth referred to as design space $P=[x_1, x_2, ..., x_n]$) which represent possible candidates in the optimization problem, together with a means of saying that P_1 is a "better" candidate than P2 (written $P1 \subset P2$) for any two points in the space. There is presumably a single point P^*, the solution, with the property $P^* \subset P$ for all $P \neq P^*$. Algorithm 1 explains the direct search method.

Algorithm 1 Direct Search Method [3]

```
1.   Select a point $B_o$ arbitrarily as the first "base point"
2.   i=1
3.   repeat
4.       Select a new point $P_i$
5.       if $P_i \subset B_{i-1}$ then
6.           $B_i = P_i$
7.       else
8.           $B_i = B_{i-1}$
9.       end if
10.      $i = i + 1$
11.  until No better points are found
     Ensure: $P^* = B_i$
```

In the above algorithm one of the key steps is selecting a new point. Based on the strategy for choosing a new point, direct search methods can be classified into different categories:

- Random Search
- Pattern search methods
- Simplex based methods
- Methods with adaptive sets of search directions

Each class of methods defines a basic idea or strategy for finding the new point in the space. Following methods are discussed with minimization of objective function as the optimization problem to be solved.

2.1 Random Search:

2.1.1 One-at-a-time search:

One-at-a-time search method is also known as alternating variable method from it's form in two dimensions [15]. This is the simplest strategy which consists of minimizing with respect to each independent variable in turn. As shown in figure (1), for two dimensional case, first one variable is varied until no further improvement can be obtained, then the next one and this sequence is repeated with ever-decreasing steps.

One of the drawback of this method is that in most practical cases where the direction of optimum is not along any coordinate axes, the progress is slow and it becomes very inefficient as the number of variables increase [3].

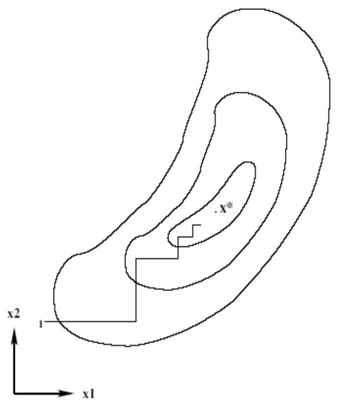

Figure (1): One-at-a-time search method for function of two variables[3]

2.2 Pattern search methods

Pattern search methods try to find a “better” search direction than simple directions along coordinate axes as in random search methods. This better search direction is found using exploration in the design space. The procedure of going from one point to the new point in design space is called a move. A move is termed a success if the value of $f(P_{i+1})$ is less. Than $f(P_i)$.; otherwise, it is a failure [3].

2.2.1 Hooke and Jeeves pattern search

The pattern search method as described by Hooke and Jeeves (referred mostly as the pattern search method in literature) makes use sequence of exploratory moves and pattern moves.

1. **Exploratory move:** In exploratory move each coordinate direction is examined in turn in the following way. A single step is taken along the direction (i) (by adding an increment Δ to variable x_i). If the move is successful, then the new value of the variable is retained. If the step fails step is taken in opposite direction (by subtracting Δ from variable xi). If this move is successful then that value of variable is retained otherwise the original value of x_i is kept.
When all the (n) coordinate directions have been investigated the exploratory move is complete. The point arrived at as a result of this procedure, which may or may not be distinct from the point from which the move originated, is called the base point.

2. **Pattern move:** Initial base point and the base point obtained using the exploratory move define the “pattern” or the search direction. Pattern move takes a single step from present base point in the direction specified by the pattern. This becomes the new starting point for next exploratory move [3].

When a pattern move and successive exploratory move fail, the algorithm returns to the previous base point. If the exploratory move around this base point also fails the pattern is destroyed and increment Δ is reduced. The whole algorithm is repeated starting from this point. The search is terminated when the increments fail below prescribed limit.

As shown in figure (2) point P_1 (marked 1) is the first base point B_o. First exploratory move from B_o begins by incrementing x_1 and resulting in P2. Since f(P2) < f(P1), P2 is retained and exploration is continued by incrementing x_2. f(P3) < f(P2) so P3 is retained in place of P2. The exploratory move is complete and P3 becomes the second base point B1. Pattern move is made in the direction of B1 −B_o from P3 to P4 (B1 −B0 = P4 −P3). Now f(P4) is not computed, but an exploratory move is performed to improve on the pattern direction. The best point found along x1 coordinate is P5. Since the second move along x2 fails, as the points obtained (P6 and P7) are not better than P5, exploratory move is complete and P5 is retained. As f(P5) < f(B1) = f(P3), it becomes the new base point B2.

Similarly the next base point B3 is obtained as P10. Now a pattern move is made to point P11. Subsequent exploratory move tries points P12, P13, P14, P15 and fails, so we come back to P10. Since f(P11) $\geq$ f(P10), pattern move itself has failed and we come back to the previous base point at P10. Fresh set of exploration to points P16, P17, and P18 also fail, causing the pattern to be destroyed and increment Δ to be reduced. The whole procedure is restarted at point P10.

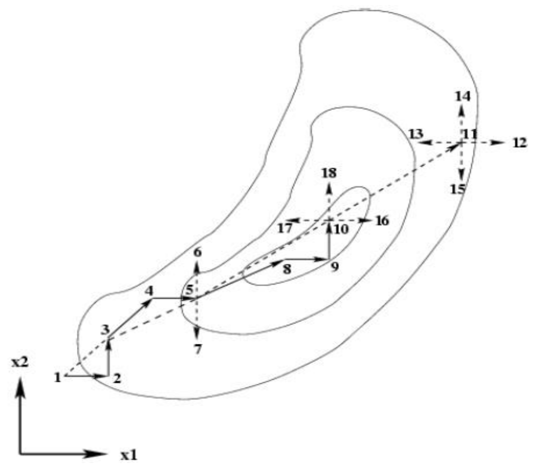

Figure (2): Hook and Jeeves Pattern Search for two dimensions [3]

2. Parallel direct search methods:

The Parallel direct search methods are designed to solve the unconstrained minimization problem: $\min_{x \in R^n} f(x)$, where f: $R^n \rightarrow R$.

What distinguishes the direct methods from other optimization methods is that they require only that the function f be continuous.

Direct search methods neither require nor estimate derivatives. As a consequence, while they are usually slower to converge than derivatives based methods, they are usually much more robust in situations where the function value are subject to noise, analytic derivatives are unavailable, or finite difference approximations to gradient are unreliable. Furthermore the direct search schemes given here parallelize very well, although they can certainly be used as sequential methods [14].

3.3.1 The first parallel algorithm:

It is clear from the steps of the of the direct search methods algorithm that they are independent processes. Hence each function evaluations process can be carried out in a processor of a MIMD computer. The number of the processors which are used is 2n+1 (n represents the number of the variables).

We can calculate the speed-up factor (Sp) of the parallel algorithm by:

$$S_p = \frac{\text{Total execution time of the task in a single processor}}{\text{Total time of the execution of the task in p processors}}$$

It is also clear that we can find $f(x_1+h,x_2,..,x_n)$, $f(x_1-h,x_2,..,x_n)$, $f(x_1,x_2+h,..,x_n)$, $f(x_1,x_2-h,..,x_n)$,…, $f(x_1,x_2,..,x_n+h)$, $f(x_1,x_2,..,x_n-h)$ in independent processors, so that they can be calculated simultaneously. Comparisons between $f(x_1+h, x_2,..,x_n)$ and $f(x_1,x_2,..,x_n)$, $f(x_1-h,x_2,..,x_n)$ and $f(x_1,x_2,..,x_n)$, $(x_1,x_2+h,..,x_n)$ and $f(x_1,x_2,..,x_n)$, $(x_1,x_2-h,..,x_n)$ and $f(x_1,x_2,..,x_n)$, …, $f(x_1,x_2,..,x_n+h)$ and $f(x_1,x_2,..,x_n)$, $f(x_1,x_2,..,x_n-h)$ and $f(x_1,x_2,..,x_n)$ are independent so that they can be carried out in parallel.

For simplicity, we can assume $P = \{x_1, x_2,…, x_n\}$, $\Delta x_i = x_i+h$, $\nabla x_i = x_i-h$, and the first parallel algorithm becomes as follows:

Algorithm 2 Parallel Direct Search Method

```
1. Choose an initial point ($P_0$)
2. Choose an initial step size h
3. Do in parallel:
o   CPU1: Find $f(\Delta P_1)$
If $f(\Delta P_1)<f(P_1)$
$x_1=x_1+h$ ; send $x_1$ to $CPU_{2n+1}$
else
get new ($P_1$) from $CPU_{2n+1}$
o   CPU2: Find $f(\nabla P_1)$
If $f(\nabla P_1)<f(P_1)$
$x_1=x_1-h$ ; send $x_1$ to $CPU_{2n+1}$
else
get new ($P_1$) from $CPU_{2n+1}$
o   CPU3: Find $f(\Delta P_2)$
If $f(\Delta P_2)<f(P_2)$
$x_2=x_2+h$ ; send $x_2$ to $CPU_{2n+1}$
else
get new ($P_2$) from $CPU_{2n+1}$
o   CPU4: Find $f(\nabla P_2)$
If $f(\nabla P_2)<f(P_2)$
$x_2=x_2-h$ ; send $x_2$ to $CPU_{2n+1}$
else
get new ($P_2$) from $CPU_{2n+1}$
.
.
.
o   CPU2n-1: Find $f(\Delta P_1)$
If $f(\Delta P_n)<f(P_n)$
$x_n=x_n+h$ ; send $x_n$ to $CPU_{2n+1}$
else
get new ($P_n$) from $CPU_{2n+1}$
o   CPU2n: Find $f(\nabla P_n)$
If $f(\nabla P_n)<f(P_n)$
```

$x_n=x_n-h$; send x_n to CPU_{2n+1}
else
get new (P_n) from CPU_{2n+1}

4. On CPU2n+1: Check If one of these steps yields to a smaller f(P) $\Rightarrow$ new iterate (P*).
5. Otherwise: try again with a step half as long (h=h/2).
6. As (P^*) approaches the solution, the algorithm reduces the length of the steps.
7. Stopping criteria: step length falls below a certain tolerance.

Parallel method can be described in the following shape:

CPU1
```
Find f(ΔP_1)
If f(ΔP_1)<f(P_1)
    x_1=x_1+h
    send x_1 to CPU_2n+1
else
    get new (P*) from CPU_2n+1
```
CPU2
```
Find f(∇P_1)
If f(∇P_1)<f(P_1)
    x_1=x_1-h
    send x_1 to CPU_2n+1
else
    get new (P*) from CPU_2n+1
```
CPU3
```
Find f(ΔP_2)
If f(ΔP_2)<f(P_2)
    x_2=x_2+h
    send x_2 to CPU_2n+1
else
    get new (P*) From CPU_2n+1
```
CPU4
```
Find f(∇P_2)
If f(∇P_2)<f(P_2)
    x_2=x_2-h
    send x_2 to CPU_2n+1
else
    get new (P*) from CPU_2n+1
```
…

CPU 2n-1
```
Find f(ΔP_n)
If f(ΔPn)<f(Pn)
    x_n=x_n+h
    send x_n to CPU_2n+1
else
    get new (P*) from CPU_2n+1
```
CPU 2n
```
Find f(∇Pn)
If f(∇P_n)<f(P_n)
    x_n=x_n-h
    send x_n to CPU_2n+1
else
    get new (P*) from CPU_2n+1
```

↑↓ ↑↓ ↑↓

CPU2n+1
```
Receive the new values from CPU1 .. CPU2n
If no reduction on the function values, then h=h/2, send h to CPU1.. CPU2n
Else send new values P* to CPU1 .. CPU2n
Until No better points are found
```

Figure (3): Tasks distribution on processors of the first parallel algorithm

3.3.2 The second parallel algorithm:

We can decrease the no. of CPUs used in the first method to reduce the cost or to use when the no. of the variables $(x_1, x_2,\ldots, x_n)$ is less. In this algorithm each function evaluations process can be assigned to one of the processors of a MIMD computer, which consists of n+1 processors.

The first step of CPU1 calculates $f(\Delta P_1)$ and compares it with $f(P_1)$. If $f(\Delta P_1)<f(P_1)$ then considering $x_1 = x_1+h$, otherwise $x_1 = x_1-h$, the same procedures are applicable on the CPU2 .. CPUn. In the next step, CPUn+1 receives the new values from CPU1..CPUn, if there is no new value then we try with a step half as long (h=h/2) and send h to CPU1..CPUn, else send the new values to CPU1 .. CPUn, till we find no reduction in reduction in the function value.

Parallel method can be described in the following shape:

CPU1
```
Find f(ΔP_1)
If f(ΔP_1)<f(P_1)
    x_1=x_1+h
    send x_1 to CPU_2n+1
Else    x_1=x_1-h
    send x_1 to CPU_2n+1
get new (P_1) from CPU_2n+1
```
CPU2
```
Find f(ΔP_1)
If f(ΔP_2)<f(P_2)
    x_2=x_2+h
    send x_2 to CPU_2n+1
Else    x_2=x_2-h
    send x_2 to CPU_2n+1
get new (P_2) from CPU_2n+1
```
…

CPUn
```
Find f(ΔP_n)
If f(ΔP_n)<f(P_n)
    x_n=x_n+h
    send x_n to CPU_2n+1
Else    x_n=x_n-h
    send x_n to CPU_2n+1
get new (P_n) from CPU_2n+1
```

↑↓ ↑↓ ↑↓

CPUn+1

```
Receive the new values from CPU1 .. CPUn
If no reduction on the function values, then h=h/2, send h to CPU1.. CPUn
Else send new values to CPU1 .. CPUn
Until No better points are found
```

Figure (4): Tasks distribution on the processors for the second parallel algorithm

5. Comparison between the suggested parallel methods:

The results of study showed that there are many differences between the two methods. These differences can be summarized in the following table:

Table (3) Differences between the suggested parallel algorithm

	First Parallel Method	**Second Parallel Method**
1)	We need 2n+1 processors whatever the number of variables	We need n+1 processors. n is the number of variables
2)	The program takes less time than the second algorithm when the variables of the function increase..	The program takes more time than the first algorithm when the variables of the function increase.
3)	**It is costly for solving high dimensional problems. it is preferred to use a such method in simple problems**	It is not costly for complex problems because all problems need n+1 processors.

REFERENCE

[1]. AL-Murshid H. (2000) An investigation of parallel numerical algorithms for solving stiff ODEs suitable for parallel computers, Ph.D. Thesis, College of Computer and mathematical Science Mosul University.

[2]. Al-Wajih K. (1999), Parallel algorithms for solving Unconstraint Optimization problems, M.Sc. Thesis , College of Computer and mathematical Science, Mosul University.

[3]. Amitay Isaacs (2003) "Direct-Search Methods and DACE", Ph.D. Thesis, Indian Institute of Technology, India.

[4]. Brocklehurst E. (1992) Parallel processing, NPL News, No. 372, Pp.24-27.

[5]. Audetand C, Dennis J. E. Jr., (2003), Analysis of generalized

pattern searches, SIAM J. Opt., (13), Pp: 889-903.

[6]. Flynn M. (1972), Some computer organizations and their effectiveness,

IEEE Trans. On Computers, Vol.C-21, No.9 pp. 948-960.

[7]. Jacoby S. L. S., Kowalik J. S. and Pizzo J. T., (1972) Iterative Methods for Nonlinear Optimization Problems, Prentice-Hall., Englewood Cliffs, New Jersey.

[8]. Khalaf B. (1990) , Parallel numerical algorithm for solving ODEs, Ph.D. Thesis, School of Computer Studies, Leeds University.

[9]. Khalaf B. (1995), A classification for computer architectures J.Educ. Sci, Vol. (24), pp. 212-222.

[10]. Kolda, T. G., Lewis, M. R., Torczon, V. (2003): Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods, SIAM Review, Volume 45, Number 3, pp. 385-482.

[11]. Meiko (1989), A programmer introduction to Sun- CSTools, Meiko LTD, England.

[12]. Rangarajan K. Sundaram. (1996), "A First Course in Optimization Theory", Cambridge University Press.

[13]. Robert Hooke and T. A. Jeeves. (1961), Direct search solution of numerical and statistical problems. Journal of the Association of Computing Machinery, pp: 221–224.

[14]. Virginia Torczon (1989), Multi-directional search: A Direct Search Algorithm For Paralel Machines, Ph.D Thesis, Department of Mathematical Science,Rice University,Houston, USA.

[15]. Murray W., (1972). "Numerical Methods for Unconstrained Optimization". Academic Press.