Parallelization of the Discrete Gradient Method of Non-smooth Optimization and Its Applications

G. Beliakov[1], J.E. Monsalve Tobon[2], and A.M. Bagirov[2]

[1] School of Information Technology, Deakin University, 221 Burwood Hwy. Burwood 3125, Australia, gleb@deakin.edu.au
[2] Centre for Informatics and Applied Optimization
School of Information Technology and Mathematical Sciences
The University of Ballarat, Victoria 3353, Australia
esteban@caribe.apana.org.au,a.bagirov@ballarat.edu.au

Abstract. We investigate parallelization and performance of the discrete gradient method of nonsmooth optimization. This derivative free method is shown to be an effective optimization tool, able to skip many shallow local minima of nonconvex nondifferentiable objective functions. Although this is a sequential iterative method, we were able to parallelize critical steps of the algorithm, and this lead to a significant improvement in performance on multiprocessor computer clusters. We applied this method to a difficult polyatomic clusters problem in computational chemistry, and found this method to outperform other algorithms.

1 Introduction

Numerical optimization is a generic task in the core of many models in science. One important example comes from theoretical chemistry, where a potential energy of a molecule as a function of individual atom positions, needs to be minimized to find the most stable conformation of this molecule. Instances of this problem appear in studies of molecular conformations, protein folding, polyatomic clusters, etc. [11,13,15]. One characteristic feature of many such models is very complicated objective function, which usually comes in the form of a black-box (i.e., a third party proprietary software), and very few assumptions can be made about it. Derivatives, even if they formally exist, are extremely difficult to express analytically, since most molecular descriptions involve some internal sets of coordinates, not readily expressed in terms of optimization variables [1, 7,11]. Moreover, not all models involve smooth functions. It is therefore quite important to have very robust, effective derivative free optimization algorithms, which do not rely on assumptions about certain properties of the objective function (such as differentiability), as these may turn out to be wrong. In this paper we study one such method, called Discrete Gradient (DG) [3,4].

The typical number of variables in molecular conformation problems ranges from several dozens to several hundreds, and the potential energy surface (PES)

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2659, pp. 592–601, 2003.

is very rugged, involving multiple local minima and other stationary points. This makes optimization procedure computationally very expensive, which dictates the need for parallelization [9].

Multiprocessor clusters are a viable (and inexpensive) alternative to traditional supercomputers. The gain in performance is achieved by executing parts of the algorithm in parallel on different processors, and then merging the results. There are tools for automatic parallelization of serial program code, however given that optimization methods based on descent are serial in their very nature (i.e., iteratively moving from one point to a better one along the direction of descent), these tools are likely to fail in such cases. It is necessary to have a good understanding of the algorithm to identify those few opportunities that can help improve performance of serial algorithms.

After briefly describing the main ideas behind the Discrete Gradient method (for in-depth mathematical treatment refer to [3,4]), we discuss the ways in which parts of this method can be parallelized, and their limitations. We will then present results of some numerical experiments, which confirm the success of our strategies. In the last section we discuss application of DG to one benchmark problem in computational chemistry, that of polyatomic clusters. We will briefly introduce this application, show how it can be solved using DG, and compare our results to previous approaches.

2 Discrete Gradient Method

Non-smooth optimization is an important area of mathematical programming. As the name implies, no assumptions about differentiability of the objective function are made. This is frequently the case where the objective function f is piecewise continuously differentiable or given as a black-box, possibly as a solution of another complicated problem. We only assume Lipschitz continuity of f.

Many methods of smooth optimization, when all involved functions are assumed to be continuously differentiable or twice continuously differentiable, are based on gradient, Hessian or their various approximations. In nonsmooth optimization functions involved are no longer continuously differentiable. Nonsmooth analysis is a theoretical basis for nonsmooth optimization. Comprehensive description of nonsmooth analysis can be found, for example, in [5,6]. The notion of a subgradient is one of the key notions in nonsmooth analysis. The subgradient is a generalization of the notion of a gradient for Lipschitz continuous functions. The set of subgradients is called a subdifferential. Different generalizations of gradient were proposed and studied by many authors. Two of them: the Clarke subdifferential and Demyanov-Rubinov quasidifferential are widely used. In this paper we will consider a version of the discrete gradient method which based on the approximations to the quasidifferential. First we recall the definition of the quasidifferential.

Let f be a Lipschitz continuous function defined on an open set $X \subset \mathbb{R}^n$, where $\mathbb{R}^n$ is n-dimensional Euclidean space. This function is called directional

differentiable at a point $x \in X$ if the limit

$$f'(x,g) = \lim_{\alpha \to +0} \alpha^{-1}[f(x+\alpha g) - f(x)]$$

exists for any $g \in \mathbb{R}^n$. The function f is called quasidifferentiable at a point $x \in X$ if it is directionally differentiable and there exist compact, convex sets $\underline{\partial} f(x)$ and $\overline{\partial} f(x)$ such that

$$f'(x,g) = \max_{v \in \underline{\partial} f(x)} \langle v, g \rangle + \min_{w \in \overline{\partial} f(x)} \langle w, g \rangle.$$

Here $\langle \cdot, \cdot \rangle$ stands for a scalar product in $\mathbb{R}^n$. The pair $Df(x) = [\underline{\partial} f(x), \overline{\partial} f(x)]$ is called a quasidifferential of the function f at a point x. The set $\underline{\partial} f(x)$ is said to be a subdifferential and the set $\overline{\partial} f(x)$ a superdifferential of the function f at a point x.

Let a function f be quasidifferentiable at $x \in \mathbb{R}^n$. For point x to be a minimum point of f on $\mathbb{R}^n$ it is necessary that

$$-\overline{\partial} f(x) \subset \underline{\partial} f(x). \tag{1}$$

A point $x \in \mathbb{R}^n$ satisfying (1) is called an inf-stationary point of the function f on $\mathbb{R}^n$. If $x \in \mathbb{R}^n$ is not an inf-stationary point then

$$-\overline{\partial} f(x) \not\subset \underline{\partial} f(x)$$

and

$$\max_{w \in \overline{\partial} f(x)} \min_{v \in \underline{\partial} f(x)} \|v + w\| > 0.$$

Then the direction $g^0 = -\|v^0 + w^0\|^{-1}(v^0 + w^0)$ where

$$\|v^0 + w^0\| = \max_{w \in \overline{\partial} f(x)} \min_{v \in \underline{\partial} f(x)} \|v + w\| > 0,$$

is a direction of steepest descent. Thus, we need to compute the entire quasidifferential of the function f to define the steepest descent direction. However, often it is impossible.

In [4] a method for minimizing quasidifferentiable functions based on the notion of a discrete gradient is developed. The discrete gradient is a finite difference estimate to a subgradient. Unlike many other finite difference estimates to a subgradient the discrete gradient is defined with respect to a given direction which allows one to get good approximation for the quasidifferential.

In [4] an algorithm for the calculation of the descent direction of a quasidifferentiable function by using discrete gradients is proposed. This is a terminating algorithm, i.e. it calculates discrete gradients step by step, and after a finite number of iterations either the descent direction is calculated, or it is found that the current point is an approximate stationary point. In the discrete gradient method Armijo's algorithm is used for a line search [2].

The discrete gradient method proceeds as follows. At a given approximation, it calculates the descent direction by calculating the discrete gradients step by step, and improving the approximation of the Demyanov-Rubinov quasidifferential. Once the descent direction is calculated, Armijo's algorithm is used for line search. The local minimum in this direction is chosen as the next approximation. Detailed treatment of this method is in [3,4].

A key step in the discrete gradient is the calculation of the descent direction. We calculate it using approximations to the subdifferential and superdifferential. This problem is reduced to a certain quadratic programming problem which is solved repeatedly at each iteration.

One of the characteristic features of DG method is its ability to "skip" many shallow local minima of the objective function, and converge if not to the global, to a deep local minimum. This feature is particularly important for difficult multiextrema optimization problems, such as those that arise in theoretical chemistry. The number of local minima in these problems can be of order 10^{20}, most of them shallow and not important from the chemical point of view. This feature has been predicted from theoretical point of view, but in this study we empirically demonstrated that this is actually the case. We compared DG method with another derivative free local method (which does converge to the nearest local minimum), and found that DG converges to a "better" solution (see Sect. 5).

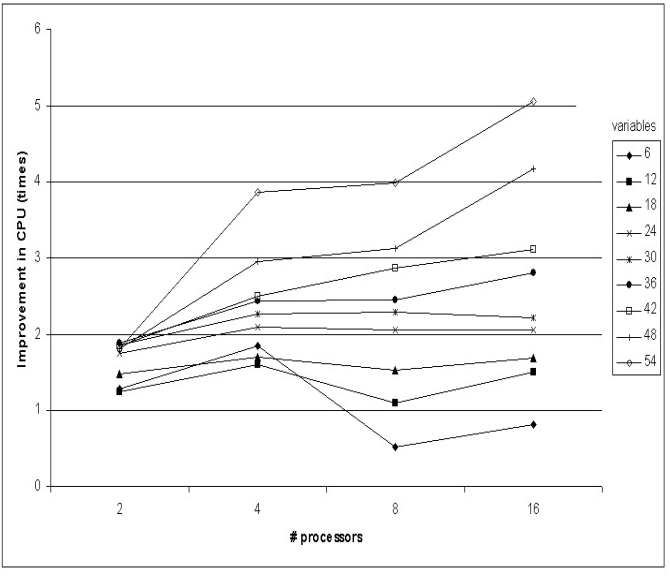

Fig. 1. Improvement in CPU time (with respect to run on a single processor) as a function of the number of processors and number of variables

3 Parallelization Strategies

The need for parallelization stems from the high computational cost of multivariate non-smooth optimization problems, and the need to solve these problems many times (e.g., as part of another algorithm). However, since the optimization strategies based on iterative descent are not parallel in their nature, there is only limited scope for parallelization. In the case of Discrete Gradient method, we identified two places, where the algorithm spends considerable time, that could be parallelized.

The way the algorithm computes the discrete gradient is by taking n values of the function (n is the number of variables), in the neighborhood of a current approximation, in the dynamically calculated directions. Then, two sets (approximations to the subdifferential and superdifferential) of points are built using these values. A subproblem of calculating the minimal distance from one set to the convex hull of the other is solved. The latter is a quadratic programming problem, and a terminating Wolfe's algorithm is used for this purpose. In degenerate cases, one of the sets may collapse into a point (the origin).

One obvious place to parallelize the algorithm is the Wolfe's algorithm [19], which solves a quadratic programming problem. A collection of such problems needs to be solved and this can be done in parallel. However, it is necessary to supply various processors with the coordinates of all the points from the two sets, and these are computed dynamically. The second opportunity to parallelize the algorithm is in computing $n+1$ values of the function in the neighborhood of the current approximation. This improvement would have a big impact on problems with complicated objective function, such as those coming from chemistry.

We implemented both parallelization strategies using the Message Passing Interface (MPI) standard [18]. Running DG algorithm on m processors, one of the processors takes the role of the main driver (master), and the rest take the role of workers (slaves). Master processor distributes the work to slaves and then merges the results. Since usually $m < n$, slaves will usually have more than one basic task to complete. These tasks (computing the value of the function and finding the shortest distance from a base point to a polytope) are not equivalent among themselves: the objective function may have different computational complexity depending on the argument, and Wolfe's algorithm for different base points may finish earlier. Therefore allocating the same number of basic tasks to slave processors is not appropriate, since the slowest processor will be the bottleneck. Instead, the slave processors query to a queue of tasks maintained by the master processor. Thus, processors that finished their tasks early are not idle, and processors with more computationally expensive tasks have fewer tasks.

Of course, the serial part of the algorithm becomes its bottleneck, and we did not hope to achieve complete parallelization of this essentially serial algorithm. However, parallelization of expensive parts of the serial algorithm yielded substantial improvements. The gain in computing time is illustrated in Fig. 1.

Of course, there is no much point in having more processors than the number of variables n, and for $m = n$, the slowest processor will dominate the speed of the algorithm. There is an optimal value for the ratio m/n, but it varies much with

the objective function. This optimal ratio can be found empirically for a class of objective functions, and subsequently used in cases where multiple optimization tasks are performed (i.e., optimizations from various initial approximations).

4 Application: Polyatomic Clusters Problem

In this section we discuss one practical application of DG, which came from theoretical chemistry. This problem is well studied, it is easy to formulate (but extremely difficult to solve), and it serves as a benchmark for optimization algorithms. Its variations are also important on their own (e.g., in studies of noble gases, interstellar dust), and as subproblems in other models, such as protein folding problem [8,10,12,16,17].

The simplest instance of this problem is the Lennard-Jones cluster problem. It consists in finding the geometry of a cluster of N identical particles loosely bound by interatomic forces. The optimal geometry minimizes the potential energy of the cluster, expressed as a function of Cartesian coordinates

$$E(x, y, z) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} v(r_{ij}) \tag{2}$$

The pairwise potential (the Lennard-Jones potential) is given by

$$v(r_{ij}) = \left(\frac{1}{r_{ij}^{12}} - \frac{1}{r_{ij}^{6}} \right) \tag{3}$$

where

$$r_{ij} = (x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2.$$

Variations of this problem include Carbon C60 clusters, water molecule clusters, An -clusters, etc., these are presented in [17].

The objective function of the Lennard-Jones problem (2) is very simple to compute, but it has extremely complicated landscape, with huge number of local minima (for the 147 atoms problem). The number of variables for an N-atom problem is $3N - 6$ (accounting for 3 translations and 3 rotations). There are at least $N!$ global minima (due to permutations of identical atoms).

Even though the objective function is smooth, the ability of DG to skip many local minima will be of great value (it is indeed the case as shown in Sect. 5). Traditional approach to this problem is to build clusters incrementally, by using the solution for $N - 1$ atom problem as the initial approximation to the N-atom problem [10,16]. Most of the optimal geometries follow this rule, however in special cases (such as $N = 38, 75, 76, 77, \ldots$) this rule is broken, and solution, say at $N = 38$ is completely different from that of $N = 37$. Hence these special "difficult" clusters essentially require random starting points and serve as benchmarks for many optimization algorithms [10].

One approach taken in [10] is to modify the Lennard-Jones potential with a penalty, favoring compact clusters

$$\bar{v}(r) = \left(\frac{1}{r^{12}} - \frac{1}{r^6} + \mu r + \beta \max\{0, r^2 - D^2\} \right) \tag{4}$$

μ, β and D are penalty parameters. The authors of [10] reported that first minimizing the modified objective function, and then using its minimum as the initial approximation to the original problem yielded convergence to the global minimum with much greater success than starting local optimization with random points (Table 1). Thus, the expected number of runs for locating the global minimum is substantially smaller. This can be projected to other instances of polyatomic clusters problem [8,10,17].

In our study we followed the penalty approach from [10], but used DG rather than other local optimization technique. The ability of DG to skip local minima paid off, and the global minima were located with higher frequency than that of [10].

5 Results

In this section we present three types of results of numerical experiments with the Discrete Gradient method. The first result relates to parallelization of the algorithm. Figure 1 presents improvements in CPU time taken by our implementation of the DG method running on the specified number of processors. As the objective function, the Lennard-Jones clusters problem (2) was taken for the specified number of variables. Ideally the improvement should be linear in the number of processors, but interprocessor communications and serial parts of the algorithm reduce it. This value can become smaller than 1 (loss in efficiency). It is clear from the plot that running DG in parallel on a number of processors is warranted only for a relatively high number of variables.

The second result illustrates the ability of DG to skip shallow local minima. We used the Lennard-Jones clusters problem with $N = 10$ (24 variables) as the objective function. The true global minimum is at $f = -28.42$. We started DG and another local derivative free method UOBYQA by M.J.D. Powell [14] from the same randomly chosen initial points, and compared the local minima both methods converged to. Figure 2 presents the plot of values at local minima of one method against the other. Points on the diagonal correspond to both methods converging to the same local minimum, whereas points above the diagonal mean the DG has found a better minimum. Figure 3 shows the frequency of locating a minimum in a given interval starting from a random point. It is clear from both plots that DG systematically converges to substantially better local minima (thus skipping the shallow minima other local methods are trapped in).

Finally, we used the modification of the objective function (3) to improve the frequency of locating the global minimum, as proposed in [10]. We ran DG from a number of random starting points and calculated the relative frequency of locating the global minimum. Table 1 compares the success rate of DG with

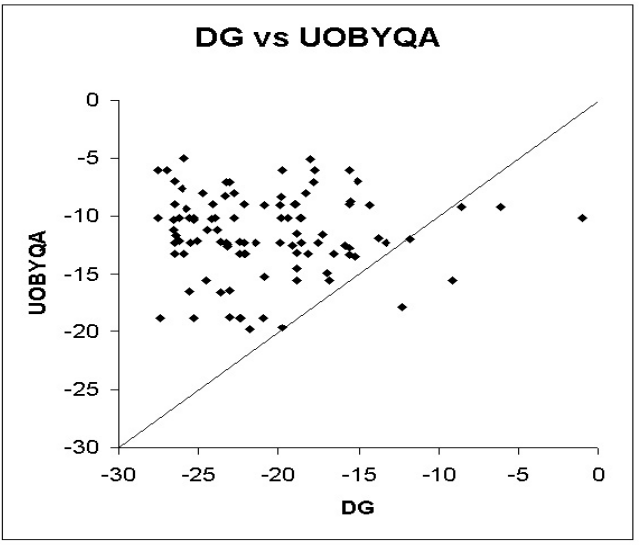

Fig. 2. Solution found by the Discrete Gradient vs. that of M.J.D. Powell's method

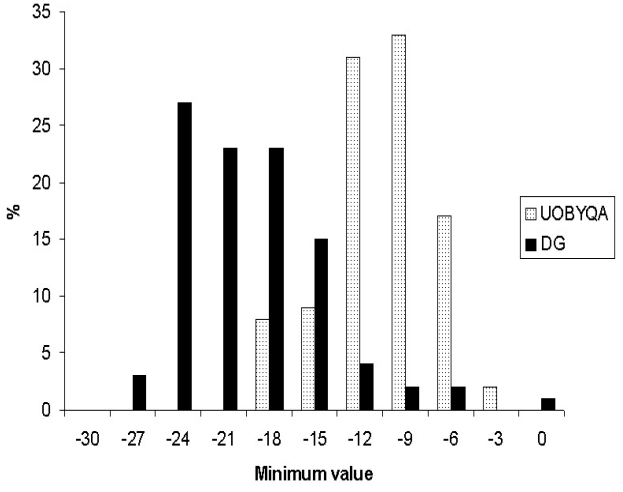

Fig. 3. Frequency of locating a local minimum from a randomly chosen initial point

Table 1. The frequency of locating the global minimum from a random initial point by DG and standard Conjugate Gradient method from [10]

N	DG Method Success Rate	CG Success Rate
11	0.25926	0.01488
12	0.78947	0.02254
13	0.20000	0.02185
15	0.06667	0.02230
16	0.19444	0.01338
17	0.05419	0.00540
23	0.10170	0.00428
38	0.41667	0.00887

that of the standard Conjugate Gradient method from [10]. Again, clearly DG is superior, since it is consistently more reliable in locating the global minimum. It is most noticeable for $N = 38$ "difficult" cluster.

6 Conclusion

Non-smooth optimization has many important applications in scientific computing. Computational chemistry provides a range of such applications related to molecular structure prediction. The objective function in such problems can have very complicated structure, or even given as a black-box, and no assumptions about differentiability can be made. The Discrete Gradient method is based on generalizations of the notion of gradient equipped with a full-scale calculus.

We have shown that parts of the DG method can be parallelized for its use on distributed memory computer clusters. Such parallelization only makes sense for a relatively large number of variables. This is the kind of problems that frequently come from computational chemistry. We examined one such problem, the Lennard-Jones clusters problem, and found that DG has an advantage over other optimization methods. It is the ability of DG to skip many shallow local minima that makes it suitable for difficult multiextrema problems.

Acknowledgements. This research was supported by the Victorian Partnership for Advanced Computing.

References

[1] M.P. Allen and D.J. Tildesley, Computer Simulations of Liquids, Oxford Science Publications, Oxford, 1990.

[2] L. Armijo, Minimization of functions having continuous partial derivatives, Pacific J. of Mathematics, 16 (1966), pp. 1–13.

[3] A. Bagirov, Derivative-free methods for unconstrained nonsmooth optimization and its numerical analysis, Journal Investigacao Operacional, 19 (1999), pp. 75–93.
[4] A. Bagirov, A method for minimization of quasidifferentiable functions, Optimization Methods and Software, 17 (2002), pp. 31–60.
[5] F.H. Clarke, Optimization and Nonsmooth Analysis, John Wiley, New York, 1983.
[6] V.F. Demyanov and A.M. Rubinov, Constructive Nonsmooth Analysis, Peter Lang, Frankfurt am Main, 1995.
[7] C.A. Floudas, Deterministic global optimization: Theory, methods, and applications, Kluwer Academic Publishers, Dordrecht, London, 2000.
[8] B. Hartke, Global geometry optimization of molecular clusters: TIP4P water, Zeitschrift fur Physikalische Chemie, 214 (2000), pp. 1251–1264.
[9] J.Y. Lee, J. Pillardy, C. Czaplewski, Y. Arnautova, D.R. Ripoll, A. Liwo, K.D. Gibson, R.J. Wawak and H.A. Scheraga, Efficient parallel algorithms in global optimization of potential energy functions for peptides, proteins, and crystals, Computer Physics Communications, 128 (2000), pp. 399–411.
[10] M. Locatelli and F. Schoen, Fast global optimization of difficult Lennard-Jones clusters, Computational Optimization and Applications, 21 (2002), pp. 55–70.
[11] A. Neumaier, Molecular modeling of proteins and mathematical prediction of protein structure, SIAM Review, 39 (1997), pp. 407–460.
[12] R.V. Pappu, R.K. Hart and J.W. Ponder, Analysis and application of potential energy smoothing and search methods for global optimization, Journal of Physical Chemistry B, 102 (1998), pp. 9725–9742.
[13] P.M. Pardalos and C.A. Floudas, Optimization in computational chemistry and molecular biology : local and global approaches, Kluwer Academic Publishers, Boston, 2000.
[14] M.J.D. Powell, UOBYQA: unconstrained optimization by quadratic approximation, Mathematical Programming, 92 (2002), pp. 555–582.
[15] H.A. Scheraga, The Multiple Minima Problem in Protein Folding, Polish Journal of Chemistry, 68 (1994), pp. 889–891.
[16] D.J. Wales and D.J., Global optimization by basin-hopping at the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms, J. Phys. Chem., 101 (1997), pp. 5111–5116.
[17] D.J. Wales and H.A. Scheraga, Review: Chemistry – Global optimization of clusters, crystals, and biomolecules, Science, 285 (1999), pp. 1368–1372.
[18] B. Wilkinson and C.M. Allen, Parallel programming: techniques and applications using networked workstations and parallel computers, Prentice Hall, Upper Saddle River, N.J., 1999.
[19] P.H. Wolfe, Finding the nearest point in a polytope, Math. Progr., 11 (1976), pp. 128–149.