PARALLEL NONMONOTONE DERIVATIVE FREE ALGORITHM FOR BOUND CONSTRAINED OPTIMIZATION

Ubaldo García-Palomares, Ildemaro García-Urrea

Procesos y Sistemas   Cómputo Científico y Estadística
Universidad Simón Bolívar, Apartado 89000, Caracas 1080, Venezuela
garciap@usb.ve   ijurrea@cantv.net

Keywords: non monotone, derivative free, bound constrained

1. PRELIMINARIES

This paper presents a parallel iterative algorithm for solving the bound constrained optimization problem **(BCOP)**

$$\min_{x\in \mathcal{F}} f(x),\ \mathcal{F} = \{x \in I\!\!R^n : s \le x \le t\},$$

where the vector inequalities $s \le x \le t$ hold component wise, i.e., $s^k \le x^k \le t^k, k = 1,\dots,n$ and $f(\cdot) : \mathcal{F} \to I\!\!R$ is a real valued function of n variables with inaccurate or absent derivative information. Starting with $x_0 \in \mathcal{F}$, the algorithm generates a sequence $\{x_i\}_1^\infty \subseteq \mathcal{F}$ that, under suitable assumptions, possesses a subsequence $\{x_i\}_{i\in I}$ converging to a point $x_* \in \mathcal{F}$ satisfying a necessary condition for optimality linked to the differentiability properties of $f(\cdot)$. A salient feature of our *parallel* algorithm is that it exhibits a *fault tolerance* fixed by the user, say π; i.e., the algorithm still works even if π processors are idle or faulty at the same time.

The parallel algorithm is a natural outgrowth of previous sequential algorithms for unconstrained optimization problems, which assume that a numerical approximation of derivatives is unreliable. The forerunner derivative free algorithm was introduced by García and Rodríguez (1). Later García et al (2) suggested the non monotone version to deal with global optimization. A key concept needed for the convergence analysis of these algorithms is the generation of a set of r unit directions $D_i = \{d_{ik} \in I\!\!R^n, k = 1,\dots,r\}$ that positively spans $I\!\!R^n$; that is, any $x \in I\!\!R^n$ can be represented as a non negative linear combination of elements in D_i. For solving (BCOP) the set $D_i = \{\pm e_1,\dots,\pm e_n\}$ of unit vectors along the axis positively spans $I\!\!R^n$ and has been suggested in previous works (1; 2; 3). Numerical experiments with unconstrained problems reveal that this choice in general deteriorates the algorithm's performance. Therefore, this paper suggests a new scheme to form D_i that takes into account the geometry of the constrained region, which seems to be necessary to prove convergence (4).

2. ALGORITHM

Due to space limitations this section describes a simplified implementation of the algorithm (table 1) and outlines the convergence proof. Complete details will be given in the full length version of this paper. There are, say p processors, with a common memory accessible by them all, where the best estimate $z, f(z)$ is saved. The j-th processor fetches this information around every Γ^j seconds. Besides, the j-th processor has the following handy information at the i-th iteration:

$K_i = \{k : s^k + \delta \le x_i^k \le t^k - \delta\}, \delta > 0$	
$P_j \subseteq \{1,\dots,n\}$	variables pertaining to j
$\tau_{ij} > 0,$	radius of search
$\gamma_{ij} \ge 1,$	expansion factor
$\mu_{ij} < 1,$	contraction factor
$x_{ij} \in \mathcal{F},$	solution estimate
$\varphi_{ij} \ge f(x_{ij}),$	upper bound of $f(x)$

P_j is an index set of those components of x that can be modified by processor j. In fact, starting at any x_{ij} the j-th processor *attempts to* solve the BCOP on the subspace generated by the unit vectors $e_k, k \in P_j$; i.e. it tries to solve

$$\min_{x\in \mathcal{C}} f(x),\ \mathcal{C} = \{x \in \mathcal{F} : x^k = x_{ij}^k, k \notin P_j\}.$$

The algorithm ensures convergence to $x_* \in \mathcal{F}$ if $\bigcup_{j=1}^{p} P_j = \{1,\dots,n\}$. When $P_j = \{1,\dots,n\}$ for $j = 1,\dots,p$, the algorithm simply uses each

D_{ij} spans positively the subspace spanned by $e_k, k \in (P_j \cap K_i)$.
$D_{ij} = D_{ij} \cup \{\pm e_k : k \in (P_j \cap \neg K_i)\}$.
success = false
for $d \in D_{ij}$
  $y =$ median$(s, x_{ij} + \tau_{ij} d, t)$
  if $f(y) \leq \varphi_{ij} - 0.01(\tau_{ij})^2$
    $x_{i+1,j} = y;\ \tau_{i+1,j} = \min(\tau, \gamma\tau_{ij})$
    success= true; break
if success= false
  $x_{i+1,j} = x_{ij};\ \tau_{i+1,j} = \mu\tau_{ij}$
Update $\varphi_{i+1,j}$
if time Γ^j to retrieve z is surpassed
  if $f(x_{i+1,j}) < f(z)$
    $z = x_{i+1,j}$
  elseif $f(z) \leq \varphi_{i+1,j} - 0.01(\tau_{i+1,j})^2$
    $x_{i+1,j} = z$
    $\tau_{i+1,j} = \min(\tau, \gamma\tau_{ij});\ \varphi_{i+1,j} = f(z)$

Table 1. i-th iteration, j-th processor

processor for solving BCOP. Although highly inefficient, let us point out that we have a fault tolerance $\pi = p - 1$. We could distribute the components $1, \ldots, n$ in such a way that any q processors randomly taken may modify all components; in which case $\pi = p - q$.

We say that x_{ij} is blocked by τ_{ij} if

$$[d \in D_{ij}] \Rightarrow f(y) > \varphi_{ij} - 0.01(\tau_{ij})^2,$$

where $y =$ median$(s, x_{ij} + \tau_{ij} d, t)$. We observe that the upper bound φ_{ij} influences the performance of the algorithm significantly. Large values allow to succeed (success= true in table 1) more often and ease the *hill climbing* ability of the algorithm; on the other hand, the closer the value of φ_{ij} is to $f(x_{ij})$ the more similar the behaviour of the algorithm is to its monotone version and it might converge to the closest local minimum.

Convergence theorem. We need the following assumptions:

A1: $f(\cdot)$ is bounded below on $\mathcal{F}$, and $\{x_i\}_1^\infty$ remains in a compact set.

A2: $f(x_{ij}) \leq \varphi_{ij}; \varphi_{i+1,j} \leq \varphi_{ij}$.

Let $I \subseteq \{1, \ldots, \}$ and let i, k be two subsequent elements in I; then $\varphi_{kj} \leq \varphi_{ij} - 0.01\tau_{kj}^2$.

A3: $D_i \to D$, and D spans positively $\mathbb{R}^n$.

Let x_* be a limit point of blocked points and let $B(x_*, \rho)$ be a ball around it. If $f(\cdot)$ is convex in B with smooth directional derivatives $f'(x, d)$, then $f'(x_*, d) \geq 0$ for all feasible directions $d \in D$. Moreover, if $f(\cdot)$ is strictly differentiable at x_*, then $\nabla f(x_*)^T d \geq 0$ for all feasible directions $d \in D$.

Remark. If $K_* = \{k : s^k + \delta \leq x_*^k \leq t^k - \delta\} = \{1, \ldots, n\}$, the algorithm solves an unconstrained optimization problem and the theory developed in (1; 2) is valid. The proof of convergence is based on this fact, but it is rather lengthy and technical. It is omitted in this extended abstract.

3. CONCLUSIONS

We have sketched an algorithm for solving the Box Constraint Optimization Problem, which shares many important properties of its counterparts in unconstrained optimization; mainly *i:* It can deal with noisy functions, *ii:* Convergence for smooth convex functions, and for strictly differentiable functions is ensured under rather weak conditions, *iii:* It is non monotone, and may scape from local minima; and finally, *iv:* practical versions can be easily implemented in a multiprogramming environment with a fault tolerance fixed by the user.

REFERENCES

[1] García-Palomares, U.M. and J.F. Rodríguez (2002): New Sequential and Parallel Derivative free Algorithms for Unconstrained Optimization. SIAM Journal on Optimization, vol. 13, 79-96.

[2] García-Palomares, U.M., F.J. González-Castaño and J.C. Burguillo-Rial (2006): A Combined Global & Local Search (CGLS) Approach to Global Optimization. Journal of Global Optimization, vol. 34, 409-426.

[3] Keefer, D.L. (1973): Self-bounding Direct Search Method for Optimization. Industrial and Engineering Chemistry Process Design and Development, vol. 12, 92-99.

[4] Lewis, R.M. and V. Torczon (1999): Algorithms for Bound Constraint Minimization. SIAM Journal on Optimization, vol. 9, 1082-1099.