American Journal of Computational Mathematics, 2013, 3, 16-26
http://dx.doi.org/10.4236/ajcm.2013.31003 Published Online March 2013 (http://www.scirp.org/journal/ajcm)

A Derivative-Free Optimization Algorithm Using Sparse Grid Integration

Shengyuan Chen, Xiaogang Wang
Department of Mathematics and Statistics, York University, Toronto, Canada
Email: chensy@mathstat.yorku.ca, stevenw@mathstat.yorku.ca

Received October 2, 2012; revised November 12, 2012; accepted December 1, 2012

ABSTRACT

We present a new derivative-free optimization algorithm based on the sparse grid numerical integration. The algorithm applies to a smooth nonlinear objective function where calculating its gradient is impossible and evaluating its value is also very expensive. The new algorithm has: 1) a unique starting point strategy; 2) an effective global search heuristic; and 3) consistent local convergence. These are achieved through a uniform use of sparse grid numerical integration. Numerical experiment result indicates that the algorithm is accurate and efficient, and benchmarks favourably against several state-of-art derivative free algorithms.

Keywords: Nonlinear Programming; Derivative Free Optimization; Sparse Grid Numerical Integration; Conditional Moment

1. Introduction

Derivative-based methods can be very efficient and have been widely used in solving optimization problems. In many applications, however, the derivative of an objecttive function might be unavailable, unreliable, or very costly to compute. Many scientific and engineering optimization problems fall into this category [1]. For example, in the helicopter rotor blade design problem [2], the objective function can only be evaluated by very expensive simulation. Similar problems include the nonlinear optimization parameter tuning problem [3], medical image registration [4], dynamic pricing [5], and community groundwater problem [6]. Therefore, derivative-free methods must be used in these situations. For these problems, not only the derivative information is not available, the function evaluation could be inaccurate or noisy most of times as well. Hence, an algorithm needs the capability to generate robust searching directions without using derivative and without overly relying on individual function evaluations.

There are several commonly used derivative-free algorithms in the literature. The classical Nelder-Mead method [7] is based on simplicies and various operations defined on these simplicies. The method evaluates the objective function on a finite number of points, and decides which operation to perform accordingly. The method is simple and able to follow the curvature of the objective function. Pattern search or directional direct search algorithm [8,9] defines a set of directions beforehand, such as positive spanning sets, positive bases or just the coordinates in its simplest form, and tests one candidate point on each of these directions in a so called polling step. An optional search step could proceed the polling step to accelerate the algorithm opportunistically, in which the algorithm makes use of some known properties, heuristics, or a surrogate model on a finite number of points. The implicit-filtering method [10] is a line search method based on the simplex gradient, which is computed based on the function values at the vertices of a simplex set. DFO method [11], Powell's method [12] and Wedge method [13] are essentially trust-region methods; however, additional steps are executed since in the derivative free optimization a quadratic or linear interpolating or regression model does not necessarily improve when the trust-region radius is reduced. For extreme problems where the mathematical structure is complex or poorly understood, other heuristic optimization algorithms such as genetic algorithms, simulated annealing, artificial neural networks, tabu-search and particle swamp are also used as methods of a last resort. An excellent discussion of several commonly used derivative-free optimization methods can be found in [14].

We point out that all these derivative-free algorithms are local optimizer, *i.e.*, only convergence to a local optimal point is guaranteed. However, on the other hand, all these algorithms are highly capable of filtering out noises, escape from inferior local optimums, and often return a fairly satisfactory solution. These widely used and successful designs clearly convey a common principle: for cases like the helicopter rotor blade design problem [2], where each function evaluation is an expensive simulation and is known to be noisy, generating robust and efficient searching directions has the highest priority. It does not mean that one can not conduct derivative-free global optimization. As a matter of fact, many derivative-free global optimization algorithms have been developed with different targeted application areas, for example, the particle swarm method, Mesh adaptive direct search (MADS) [15], DIviding RETangles (DIRECT) [16], and Multileve coordinate search (MCS) [17]. More interestingly, some of the above local derivative-free algorithms have been extended to solve global optimization problems, for example, the global Direct Search, see Section 13.3 [14] and references therein. It appears that four categories of algorithms, with or without derivative, seeking local or global optimums, all find their suitable application areas.

In this paper, our goal is to generate high-quality local optimal solutions efficiently. Hence in this paper, we are concerned with the following optimization problem:

$$\max_{x\in B} f(x), \tag{1}$$

where $f(\cdot):\mathbb{R}^d \to \mathbb{R}^+ \in C^2$, but is expensive to evaluate. In many applications, such as the helicopter rotor blade design problem [2], $f(\cdot)$ is known to be smooth, but has no analytical form, and one can only rely on expensive simulations or experiments to evaluate function values. The measurements could be inaccurate or noisy though the function $f(\cdot)$ itself is known to be smooth, see [14]. Furthermore, derivatives or reliable approximation are not available. $f(\cdot)$ could have many local optimums as well. The decision variable x is subject to a box constraint $x \in B$. For example, in the parameter tubing problem [3], B is a theoretically or empirically reasonable range of the parameters to be tuned.

Our research is motivated by Wang *et al.* [18], where the authors proposed a novel derivative free global optimization method. In this iterative method, the integral of the objective function over a local neighbourhood of each iterate is computed, which further determines the next iterate. The authors showed that if the neighbourhood size could be chosen properly at each iterate, the algorithm converges to a *global* optimum. The authors demonstrated a few examples, where the algorithm successfully found global optimums while several other commonly used derivative-free methods and even derivative-based methods failed.

We have a different view of the integration based derivative free optimization method. First of all, we use the integration to define the searching directions, rather than the iterates as did in [18]. We found that it is more efficient to use the integration to define the search direction. The integration based searching direction uses information from multiple function evaluations at strategically located points, which avoids unduly relying on the gradient information at the current iterate. This is especially important since the function evaluations are noisy, especially when obtained from simulation or physical measurements, hence the local gradient, even if available or can be approximated, has poor quality. Secondly, our focus is on generating high quality local optimum efficiently, rather than seeking a global optimum. This moderate goal actually enable us to search more effectively. Finding global optimum and proving its global optimality is known to be hard. Though [18] shows the existence of a series of neighbourhood sizes such that their algorithm can find a global optimum, the exact neighbourhood size is not given in theory, nor specified algorithmically. In this paper, we design a two stage searching strategy, which aims at achieving a good balance of global coverage and local optimality.

First, we divide the solution seeking process into two stages: the global probing stage and the local convergence stage. In the global probing stage, the algorithm search for high region of the feasible domain for a maximization problem; while in the local convergence stage, the algorithm converges to peaks of the high region found in the first stage. The first stage steps are typically large; and the second stage steps are small. We update the neighbourhood size differently in the two stages, which suites the different purposes of the two stages well.

Accordingly, we prove that in the local convergence stage the algorithm always generates locally improving directions and converges; while in the global probing stage, we are able to show that the algorithm can generate searching directions towards a global optimum but against a local attraction for the mixture model functions commonly appeared in statistics optimization.

We not only develop new theories to support our new perspective, but also design critical algorithmic components accordingly. Our new line search method is a greedy algorithm which appears to have good numerical performance. In the contrast, [18] does not use line search at all. Our strategy to select the starting point is also new. It is well-known that the starting point is important for nonlinear optimization problem. However, most nonlinear optimization algorithms, including [18], rely on user defined starting points or starting from zero mechanically. We provides an innovative approach here: we start from the center of gravity of the whole feasible region, which is again computed from integration with function values as weights. Hence the starting point is close to the global optimum.

Secondly, we use the modern sparse grid numerical integration method (Smolyak 1960) in computing the local integration. The sparse grid method is highly efficient for functions with moderate dimensions, and has been widely used in engineering, finance, atmosphere studies, see [19] the reference therein. Since the function evaluation is expensive for our problems, we need a full control of *total* function calls in our algorithm: not only in the searching process, but also in the numerical integration. We derive a new closed form formula determining exactly how many point evaluations are needed in the sparse grid method. To the best of our knowledge, only the order of number of points has been shown in the sparse grid literature. Based on our new formula, we clearly show that the number of point evaluations needed in the numerical integration increases with the dimension linearly.

The paper is organized as follows. In Section 2.1, we show that the searching direction generated from local integration is always an improving direction; on the other hand, by simply enlarging the integration area, the algorithm can generate a searching direction towards toward a area with higher average function value. We also show that for a class of functions, the algorithm could find a global optimal solution. Based on these ideas, we design a new algorithm and present it in Section 5. In Section 4, we briefly review the sparse grid method and derive a new formula count its function calls. In Section 6, we test the new algorithm and benchmark against state-of-art derivative free algorithms. Conclusion and discussion are provided in Section 7.

2. Several Theoretical Results

We first prove several theoretical results which motivated the new method we will present in the later Section 5. These results cover important aspects of a nonlinear derivative free algorithm, including our choice of the searching direction; starting point strategy; and a new closed-form formula which we developed to calculate the number of function calls in the sparse grid numerical integration at each iteration.

Searching Direction and Local Convergence

For a $x \in B$ and $r > 0$ such that $B_x(r) := [x-r, x+r] \subset B$, we define the search direction to be $y - x$, where

$$y = \frac{1}{\alpha} \int_{B_x(r)} t f(t)\,\mathrm{d}t, \text{ and } \alpha = \int_{B_x(r)} f(t)\,\mathrm{d}t. \tag{2}$$

We also call y the local center of gravity of $B_x(r)$ in this paper. We prove that $y - x$ is a locally improving direction in the next Theorem.

Theorem 2.1 *The searching direction* $y - x$ *satisfies*

$$(y-x)^T \nabla f(x) > 0, \forall r \le 1.5 d^{-1.5} \left(\max_{t \in B_x(r)} \left\| \nabla^2 f(t) \right\| \right)$$

Especially, for a quadratic $f(x) = c^T x + \frac{1}{2} x^T Q x$, *we have*

$$y - x = \frac{(2r)^{d+2}}{12\alpha} \nabla f(t), \forall r \le 1.5 d^{-1.5} \|Q\|.$$

Proof. Let $B_u := [-r, r]^n$, and by definition of the local center of gravity,

$$\begin{aligned} y &= \frac{\int_{B_u} (x+u) f(x+u)\,\mathrm{d}u}{\int_{B_u} f(x+u)\,\mathrm{d}u} \\ &= \frac{x \int_{B_u} f(x+u)\,\mathrm{d}u + \int_{B_u} u f(x+u)\,\mathrm{d}u}{\int_{B_u} f(x+u)\,\mathrm{d}u} \\ &= x + \frac{\int_{B_u} u f(x+u)\,\mathrm{d}u}{\int_{B_u} f(x+u)\,\mathrm{d}u}. \end{aligned} \tag{3}$$

Hence

$$(y-x)^T \nabla f(x) = \frac{\nabla^T f(x) \int_{B_u} u f(x+u)\,\mathrm{d}u}{\int_{B_u} f(x+u)\,\mathrm{d}u} \tag{4}$$

Since $f(\cdot) \in \mathbb{R}^+$, $\int_{B_u} f(x+u) > 0$, and the sign of $(y-x)^T \nabla f(x)$ only depends on $\nabla^T f(x) \int_{B_u} f(x+u)\,\mathrm{d}u$. In the following we first study $\int_{B_u} u f(x+u)\,\mathrm{d}u$. The second Taylor expansion of $f(\cdot)$ at x for sufficiently small u is

$$f(x+u) = f(x) + u^T \nabla f(x) + \frac{1}{2} u^T H_u u.$$

We note that $H_u := \nabla^2 f(\tilde{x}(u))$ may change with u. For brevity, we also define $g := \nabla f(x)$.

$$\begin{aligned} &\int_{B_u} u f(x+u)\,\mathrm{d}u \\ &= \int_{B_u} u \left(f(x) + u^T g + \frac{1}{2} u^T H_u u \right) \mathrm{d}u \\ &= f(x) \int_{B_u} u\,\mathrm{d}u + g \int_{B_u} u u^T\,\mathrm{d}u + \int_{B_u} u u^T H_u u\,\mathrm{d}u \\ &= \frac{1}{12} (2r)^{d+2} g + \int_{B_u} \frac{1}{2} u u^T H_u u\,\mathrm{d}u, \end{aligned} \tag{5}$$

where the last equality used $\int_{B_u} u = 0 \in R^d$ and

$$\int_{B_u} u u^T\,\mathrm{d}u = \left[\int_{-r}^{r} \int_{-r}^{r} u_i u_j \right]_{ij} = \frac{1}{12} (2r)^{d+2} I_d,$$

since $\int_{-r}^{r} \int_{-r}^{r} u_i u_j \,\mathrm{d}u_i \mathrm{d}u_j = 0, i \ne j$,

$\int_{-r}^{r} u_i^2 \mathrm{d}u_i \mathrm{d}u_i = \frac{1}{3} u_i^3 \Big|_{-r}^{r} = \frac{2}{3} r^3$, and $\int_{B_u} u_i^2 \mathrm{d}u_i \mathrm{d}u_i = \frac{1}{12} (2r)^{d+2}$.

I_d is the $d \times d$ identity matrix.

For quadratic $f(\cdot)$, $H_u = Q$, hence

$$\int_{B_u} \frac{1}{2} u u^T H_u u \mathrm{d}u = \left[\sum_{k,j} Q_{kj} \int_{B_u} u_i u_k u_j \right]_i = 0,$$

since $\int_{B_u} u_i u_k u_j \mathrm{d}u_i \mathrm{d}u_k \mathrm{d}u_j = 0$ for all possible *i*, *k*, *j* combinations. Since $f \in C^2$, a maximum of $\left\| \nabla^2 f(t) \right\|$, $t \in B_x(r)$ for all *x* in *B* exists. Hence

$$\begin{aligned} & g^T \int_{B_u} u f(x+u) \mathrm{d}u \\ & = \frac{1}{12} (2d)^n \|g\|^2 + g^T \int_{B_u} u u^T H_u u \mathrm{d}u \\ & \ge \frac{1}{12} (2r)^d \|g\|^2 - \|g\| \int_{B_u} \|u\|^3 \|H_u\| \mathrm{d}u \\ & \ge \frac{1}{12} (2r)^d \|g\|^2 - \|g\| \int_{B_u} \left(\max \|u\|^3 \right) \left(\max \|H_u\| \right) \\ & \ge \frac{1}{12} (2r)^{d+2} \|g\|^2 - \frac{1}{8} \|\bar{H}\| (2r)^{d+3} \|g\| \\ & = \frac{1}{12} \|g\| (2r)^{d+2} \left(\|g\| - 3 \|\bar{H}\| r \right). \end{aligned} \tag{6}$$

Applying this result into (4),

$$(y-x)^T \nabla f(x) \ge \frac{\frac{1}{12} \|g\| (2r)^{d+2} \left(\|g\| - 3 \|\bar{H}\| r \right)}{\int_{B_u} f(x+u)} \ge 0.$$

when $r \le \frac{1}{3} \|\bar{H}\|^{-1} \|\nabla f(x)\|$. □

Theorem 2.1 shows that the algorithm always generate an improving direction. Furthermore, the theorem shows that for a quadratic objective function, the integration based search direction is exactly the steepest descent direction. Though the analytical form of the objective function $f(\cdot) \in \mathbb{C}^2$ is unknown, our closed form expression for quadratic case clearly indicates the capability of interation based searching direction, especially it is well known that sequential quadratic approximations of C^2 function can be effective, which sheds lights on future research.

With the aboved establied improving feasible direction, we can directly apply the well established idea of *feasible direction algorithm* Theorem 10.2.7 [20]. The idea is fairly simple: given a feasible point x_k, an improving feasible direction d_k satisfying (1) $x_k + \lambda d_k$ is feasible, and (2) the objective value is better at $x_k + \lambda d_k$ for sufficiently small λ, a one dimensional optimization problem is solved to determine how far to proceed along d_k, which leads to a new point x_{k+1}, and the process is repeated.

Theorem 2.2 *Let* x_k *be feasible*, $d_k = y - x_k$, y *obtained from* (2), $x_{k+1} = x_k + \lambda^* (y - x_k)$, *where* λ^* *is optimal for the one dimensional optimization problem* $\min_\lambda f(x_k + \lambda d_k), s.t.\, x_k + \lambda d_k \in B$. *Then the sequence* $\{x_k\}$ *converges to a KKT point.*

Proof. See Theorem 10.2.7 [20]. □

While our convergence analysis can take benefit of the well established optimization theories; in the contrast, the convergence analysis of the directional direct search algorithm [8,9] relies on searching a predefined set of directions, which spans the searching space.

Our next example shows that when *r* is large, the searching direction generated by the algorithm is not necessary a locally improving direction. However, as the examples shows, it points to the global optimum.

Example 2.1 *Let* $f(x) := a_0 + a_1 x + a_2 x^2 + a_3 x^3$. *Its second order Taylor expansion is* $f(x) + gu + \frac{1}{2} H_u u^2$, $g = a_1 + 2a_2 x + 3a_3 x^2$, *and* $H_u = 2(a_2 + 3a_3 x + a_3 u)$. *It can be shown that*

$$g(y-x) = \frac{2gr^3 \left(\frac{1}{3} g + \frac{1}{5} a_3 r^2 \right)}{\int_B f(x+u)} \tag{7}$$

We are not following the steps of (6) here since H_u is known explicitly. For simplicity, let $a_3 = -\frac{1}{3}$, $a_2 = 1$, $a_1 = 0$, $a_0 = 4$. With this choice, $f(x) > 0$ on $[-2,4]$. $g = -x^2 + 2x = x(2-x)$, hence $g > 0$ for $x \in (0,2)$. For *x* in this range $g > -\frac{3}{5} a_3 r^2 = \frac{1}{5} r^2$ leads to conclusion $r < \sqrt{5}$. The bound is sharp as we see that with $x = 1$ the gradient is 1. If we choose $r = 2$ within the bound, the center of gravity for this problem is

$$y = \frac{\int_{[-1,3]} x \left(x^2 - \frac{1}{3} x^3 \right)}{\int_{[-1,3]} x^2 - \frac{1}{3} x^3} = 1.04,$$

which is bigger than *x* = 1 and aligned with the gradient *g*. However, if we choose *r* = 3, then a similar calculation shows that the center of gravity on $[-2,4]$ is 0.85, which points to region with larger objective values than the local maximum.

It is not a surprise that when *r* is large, the algorithm is able to generate searching directions towards the global optimum, since more information of the function is used. In the contrast, a gradient only reflects the local structure of the function. This observation leads to our development of a new heuristic. We use large *r* at beginning of the algorithm with an intention to explore the overall structure of the objective function. This global probing phase ends when successive iterates are close enough to each other, which implies that two iterates are in the same hump. In the next phase, we would like the algorithm to find the maximum of this hump, hence we use a very small *r* and fast update to achieve the local convergence.

3. Starting Point

We know that the starting point is very important for a nonlinear optimization problem. We propose to use the center of gravity of $f(\cdot)$ on *B* as the starting point. By its construction, the center of gravity tends to locate in a high valued neighbourhood, hence provides an effective starting point. In the following theorem, we show that this simple *heuristic* starting point strategy can always lead to a global optimum for the mixture model in statistics under some assumptions. This strategy is new and seems effective in our numerical experiments.

Theorem 3.1 *Consider a statistical mixture model*

$$f(x) = p_1 f_1(x) + p_2 f_2(x), \tag{8}$$

where $f_1(\cdot), f_2(\cdot)$ *are density functions,* $p_1, p_2 \in [0,1]$, $p_1 + p_2 = 1$. *Without loss of generality, assume* $p_1 \geq p_2$. *Assume that the global optimum* x^* *is unique and satisfies* $\nabla f(x^*) = 0$. *Define*

$$\delta^0 = \arg\max_{\delta} \left\{ \delta \mid \forall x \in N_{x^*}(\delta), \partial f_1/\partial x_i \neq 0, i = 1, \cdots, n \right\},$$

where $N_{x^*}(\delta) := \left\{ x : \left\| x - x^* \right\| \leq \delta \right\}$. If $\mu_1 \in N_{x^*}(\delta_0)$ and

$$p_2 \leq \frac{\delta^0 - \left\| \mu_1 - x^* \right\|}{\left\| \mu_1 - \mu_2 \right\|}, \tag{9}$$

where $\mu_1 = \int x f_1(x) \mathrm{d}x$ *and* $\mu_2 = \int x f_2(x) \mathrm{d}x$, *then the algorithm converges to the global optimum if started from* $\bar{x} = \int x f(x) \mathrm{d}x$.

Proof. Observe that

$$\bar{x} = \int x f(x) \mathrm{d}x = p_1 \mu_1 + p_2 \mu_2. \tag{10}$$

We then have

$$\begin{aligned} \left\| \bar{x} - x^* \right\| &\leq \left\| \bar{x} - \mu_1 \right\| + \left\| \mu_1 - x^* \right\| \\ &= \left\| p_1 \mu_1 + p_2 \mu_2 - \mu_1 \right\| + \left\| \mu_1 - x^* \right\| \\ &= p_2 \left\| \mu_1 - \mu_2 \right\| + \left\| \mu_1 - x^* \right\| \\ &\leq \delta^0 - \left\| \mu_1 - x^* \right\| + \left\| \mu_1 - x^* \right\| \\ &\leq \delta^0. \end{aligned}$$

This implies that $\bar{x} \in N_{x^*}(\delta^0)$, hence the local improving directions leads to x^* if $\bar{x} \neq x^*$. □

We remark that the above result is applicable to a broader functional class with proper normalization. For a composite function

$$g(x) = g_1(x) + g_2(x), \tag{11}$$

where $g_1(\cdot), g_2(\cdot)$ are non-constant and nonnegative functions, it can be normalized as

$$\frac{g(x)}{\int g(x)\mathrm{d}x} = \frac{\int g_1(x)\mathrm{d}x}{\int g(x)\mathrm{d}x} \frac{g_1(x)}{\int g_1(x)\mathrm{d}x} + \frac{\int g_2(x)\mathrm{d}x}{\int g(x)\mathrm{d}x} \frac{g_2(x)}{\int g_2(x)\mathrm{d}x},$$

which is precisely the mixture form $f(x) = p_1 f_1(x) + p_2 f_2(x)$ with

$$f(x) = g(x) \Big/ \int g(x) \mathrm{d}x,$$

$$p_i = \int g_1(x)\mathrm{d}x \Big/ \int g(x)\mathrm{d}x, i = 1, 2,$$

$$f_i(x) = g_i(x) \Big/ \int g_i(x)\mathrm{d}x, i = 1, 2.$$

The condition, $p_1 > p_2$ implies that $\int g_1(x)\mathrm{d}x > \int g_2(x)\mathrm{d}x$. This implies that $g_1(\cdot)$ makes larger contribution to the total integration than $g_2(\cdot)$.

4. Sparse Grid Numerical Integration

Since the algorithm relies on the numerical integration to calculate the local center of a gravity at each step and the starting point, its accuracy and efficiency is crucial to the success of our algorithm. We integrate the modern sparse grid method [21] into our algorithm. In this section, we first briefly introduce the sparse grid method. Then we present our result on this aspect: a new closed form formula for the number of function calls used in the sparse grid numerical integration. The complexity of sparse grid method itself has been studied by various authors, and the order of complexity is shown in [19]. Our result makes it explicit how many function calls are used exactly in the integration step in our algorithm. Since the sparse grid method itself is not our contribution, our introduction is terse for the purpose of explaining the necessary notations used in our derivation of the closed form formula. For details of the sparse grid method, we refer our readers to [22].

Numerical integration for univariate function (quadrature) uses the weighted sum of function values as the approximation to the integration:

$$\int_B f(t)\mathrm{d}t \approx \sum_{i=1}^{K} w_i f(t_i). \tag{12}$$

A central problem in quadrature is to decide the weights w_i and grid points t_i. In this sense, Gauss-Legender quadrature uses an optimal solution to the following optimization problem:

$$\min_{\substack{t_1, \cdots, t_K \\ w_1, \cdots, w_K}} \sup_{f \in \mathbb{P}_{2K-1}} \left| \int_B f(t)\mathrm{d}t - \sum_{i=1}^{K} w_i f(t_i) \right|. \tag{13}$$

Clearly the model minimizes the worst case approximation error for any univariate polynomial function with order up to $2K-1$. Different univariate quadrature rules have been derived.

Extension of univariate quadrature to multivariate quadrature is difficult due to the curse of dimensionality. Among various approaches, the sparse grid method uses an interesting tensor product of univariate quadratures. In the following we first outline necessary details of this tensor product, on which our analysis of number of function evaluations is based on.

We define an operator Q_j^1:

$$Q_j^1[f] = \sum_{t\in U_j} w(t) f(t), \tag{14}$$

where U_j specifies the set of evaluation points, and $w: U_j \to \mathbb{R}$ provides the corresponding weights. We note that for Gauss-Legender univariate quadrature, the cardinalities for $U_j = j$. We define a difference operator as:

$$\Delta_j^1[f] := \left(Q_j^1 - Q_{j-1}^1\right)[f]$$

$$\Delta_0^1[f] := 0.$$

Clearly the difference operator Δ_j^1 is defined on the point set $\bar{U} := U_j \bigcup U_{j-1}$, and it could also be represented as a weighted sum of function values:

$$\Delta_j^1[f] = \sum_{t\in U_j} w(t) f(t) - \sum_{t\in U_{j-1}} w(t) f(t) = \sum_{t\in \bar{U}_j} \bar{w}(t) f(t),$$

where the weight function $\bar{w}: \bar{U}_j \to \mathbb{R}$ could be calculated mechanically. We define the tensor product of difference operator as

$$\left(\Delta_{j_1}^1 \otimes \cdots \otimes \Delta_{j_d}^1\right)[f] = \sum_{t_1\in \bar{U}_{j_1}} \cdots \sum_{t_d\in \bar{U}_{j_d}} \bar{w}_{j_1}(t_1) \cdots \bar{w}_{j_d}(t_d) f(t_1, \cdots, t_d). \tag{15}$$

The *sparse grid* approach (Smolyak 1960) extends the univariate quadrature Q_j^1 to the multivariate quadrature rule $A(q,d)$ (via the intermediate difference operator Δ_j^1) as for $d \ge 2$:

$$\int_B f(t)\,\mathrm{d}t \approx A_q^d[f] := \sum_{\|j\|_1 \le q+d-1} \left(\Delta_{j_1} \otimes \cdots \otimes \Delta_{j_d}\right)[f]. \tag{16}$$

A_q^d has no approximation error [23] for polynomials in the space $\bigcup \mathbb{P}_{j_1} \otimes \mathbb{P}_{j_2} \cdots \otimes \mathbb{P}_{j_d}$, $\forall j_1 + \cdots + j_d \le q + d - 1$, if each univariate quadrature rule Q_j^1 has no approximation error for the univariate polynomial space $\mathbb{P}_j$. For smooth functions, not necessarily polynomial, the sparse grid method also achieves high accuracy [19]. Also see [22] for spare grid implementation details. In this paper, the most relevant question is how many function calls are used in a sparse grid numerical integration. Though the order of magnitude of this number is known, see Lemma 3.6 in [21] and [22], we derive a new and exact formula. The exact formula shows that the number of function calls in computing the local center of gravity is $2d+1$.

Proposition 4.1 *Using the Gauss-Legender sparse grid rule* A_2^d, *Subroutine* 2 *makes* $n = 2d+1$ *function calls.*

Proof: Applying Theorem 4.1 for the special case $q = 2$.

Theorem 4.1 *The Gauss-Legender sparse grid rule* A_q^d *uses* $N(q,d)$ *function calls, where*

$$N(q,d) = \binom{q+2d-1}{2d}, \text{ for } q \le d \tag{17}$$

Furthermore,

$$N(q+1,d) - N(q,d) = \binom{q+2d-1}{2d-1}, \text{ for } q+1 \le d \tag{18}$$

Proof. For the Gauss-Legender univariate rule Q_j^1, the number of points is $\left|U_j^1\right| = j$, and $U_j \bigcap U_i = \varnothing\ \forall i \ne j$. Furthermore, [23] shows that (combination rule):

$$A_q^d[f] = \sum_{q\le \|j\|_1 \le q+d-1} (-1)^{q-d-\|k\|_1-1} \cdot \binom{d-1}{\|k\|_1 - q} \left(Q_{j_1}^1 \otimes \cdots \otimes Q_{j_d}^1\right)[f] \tag{19}$$

Hence for $q \le d, d \ge 2$,

$$N(q,d) = \sum_{j_1\ge 1}\sum_{j_2\ge 1}\cdots\sum_{\substack{j_d\ge 1 \\ k_1+k_2+\cdots+k_d \le q+d-1}} k_1 k_2 \cdots k_d. \tag{20}$$

1) The closed formula (17) apparently holds for $d = 2$ and $q \le d$ since $N(1,2) = 1$, and

$$N(2,2) = \sum_{j_1\ge 1}\sum_{\substack{j_2\ge 1 \\ j_1+j_2\le 3}} j_1 j_2 = 5$$

2) Assume the formula (17) is valid for d and $q \le d$. We now consider the case for $d+1$.

$$\begin{aligned} N(q,d+1) &= \sum_{j_1}\cdots\sum_{\substack{j_{d+1} \\ j_1+\cdots+j_{d+1} \le q-1+n+1}} j_1 j_2 \cdots j_n j_{n+1} \\ &= \sum_{j_{d+1}=1}^{q} j_{d+1} \cdot \left(\sum_{j_1}\cdots\sum_{\substack{j_d \\ j_1+\cdots+j_d \le q+d-j_{d+1}}} j_1 j_2 \cdots j_n \right) \\ &= \sum_{j_{d+1}=1}^{q} j_{d+1} \cdot \binom{q - j_{n+1} + 2d}{2d} \\ &= \binom{q+2(d+1)-1}{2(d+1)}. \end{aligned}$$

The last step

$$\binom{q+2d+1}{2d+2} = \sum_{j_{d+1}=1}^{q} \binom{j_{d+1}}{1} \binom{q+2d-j_{d+1}}{2d} \tag{21}$$

can be proved by the following equivalence: selections of $2d+2$ balls out of $q+2d+1$ identical white balls is equivalent to: choose one ball from the first j_{d+1} balls and there are $\binom{j_{d+1}}{1}$ ways; choose the $(j_{d+1}+1)th$ ball as the second ball and color it red; choose $2d$ balls from the rest and there are $\binom{q+2d-j_{d+1}}{2d}$ ways. Total number of balls before the red could be one to q. The maximal number is q since otherwise there is less than $2d$ balls behind the red.

3) Hence the formula (17) is valid by mathematical induction.

The second half of the theorem follows this identity

$$\binom{n}{j}=\binom{n-1}{j}+\binom{n-1}{j-1}. \qquad \square$$

5. Algorithm

Algorithm 1 Sparse Grid Derivative Free Algorithm (SGDF)

```
1: global probing phase:
2:  $z = cg(B)$, see Subroutine 2
3:  $\alpha = 0.9$
4: repeat
5:     $x = z$
6:     $r = argmax\ r$, s.t. $f(x)*(2r)^d \le \alpha$,
       $[x-r, x+r] \subset B$
7:     $B_\alpha = [x-r, x+r]$
8:     $y = cg(B_\alpha)$, see Subroutine 2
9:     linesearch along $y-x$ to get $z$, see Subroutine 3
10: until $\|x-z\| < \varepsilon_1$
11: local convergent phase:
12: repeat
13:     $\alpha = 0.01\alpha$
14:     steps (4)-(10)
15: until $\alpha < \varepsilon_2$
```

As shown in Algorithm 1, it has two phases: the global probing phase and the convergent phase. The two phases differs only in the parameter α^k, and are identical otherwise. In the global probing phase, $\alpha^k = 0.9$ constantly; while in the convergent phase $\alpha^{k+1} = 0.01\alpha^k$. Both phases share same iteration steps (4)-(10), which are essentially numerical implementation of (2) with line search for additional efficiency. In the global phasing phase, the algorithm uses 90% terrain of the feasible region to determine the search direction. This provides great opportunity for the algorithm to escape from local optimum traps. When two successive iterates x_i and x_{i+1} are very close, we switch to the convergent phase, where we exponentially decay α. By focusing on a less portion of the terrain, the algorithm is able to advance to a local optimum accurately. Details of one iteration is the follows. We first approximate the radius for the given α^k through mid-point quadrature rule in the step 6. The new domain B_α is defined in the step 7. Step 8 computes the center of gravity of the new domain B_α using the subroutine 2. Step 9 conducts a linear search subroutine 3 to accelerate the algorithm.

The subroutine 2 computes the center of gravity (cg), which calls the sparse grid algorithm in its first step to generate n sparse grid points as briefly introduced in section 4. In our numerical implementation, we called the public domain Matlab package Sparse Grid Toolbox by Andreas Klimke. We note that each numerical integration uses only $2d+1$ function calls as proved in the Proposition 4.1. In the subroutine 2, the volume of the domain B_α is computed in the step 3 and is used in lieu in the following step 4. We also point out that function evaluations at grid points are computed in step (2) only once. These values are stored and used later in steps (3) and (4).

Subroutine 2: Center of gravity $cg(B_\alpha)$

```
1: Generate $n$ sparse grid points $x^i$ of the given rectangle $B_\alpha$, see Section 4
2: Compute $f(x^i), x_1^i f(x^i), \cdots, x_d^i f(x^i), i = 1,\cdots,n$
3: Compute the volume $V = \sum_{i=1}^n w_i f(x^i)$
4: Compute $cg_j = \frac{1}{V}\sum_{i=1}^n w_i x_j^i f(x^i),\ j = 1,\cdots,d$
```

In the subroutine 3 we equip a special derivative free line search for SGDF enhance its numerical efficiency. The idea is to move along the line $cg(B_\alpha)-x$ as much as possible. We observe that it is often too conservative to move to $cg(B_\alpha)$ only; instead, search along the line often brings huge performance gain. The line search is greedy: it moves forward by a fixed stepsize until the function value is worse or it hits the boundary; then it bisects the stepsize and retreats a stepsize until it finds the first feasible and improving point or until the stepsize is less than ε_3. ε_1, ε_2, ε_3 are all set at $1e-8$ level.

Subroutine 3: Derivative free line search

```
Ensure: $x, y \in \mathbb{X}, f(x) < f(y)$;
1: initialize $d = (y-x)/\|y-x\|; s = 1, y^+ = y$;
2: while $f(y^+) \ge f(y)$ and $y^+$ is feasible do
3:     $y = y^+, y^+ = y + s \cdot d$
4: end while
5: repeat
6:     $s = s/2, y^+ = y^+ - s \cdot d$
7: until $y^+$ is feasible and either $f(y^+) > f(y)$ or $s < \varepsilon_3$
8: return $\arg\max_{x,y,y^+}\{f(x), f(y), f(y^+)\}$
```

6. Numerical Tests

We compare SGDF with the state-of-art derivative free algorithms, including Pattern Search or Direct Search (with and without poll step) [24], Global Search [25], Multi-start [26], Simulated Annealing and Genetic algorithm [27]. These algorithms have mature industrial strength implementations in MATLAB Global Optimization Toolbox. We also implemented our new SGDF algorithm in MATLAB and conducted comparative study.

We use their default settings for the solvers from the

Matlab Global Optimization Toolbox. Pattern Search Algorithm uses only poll step by default. However, it is more efficient when coupled with a searching step. Hence we compare with the Pattern Search algorithms with and without the searching option. Both the Global Search Solver and the Multistart Solver needs a local solver. We followed the recommendation from Matlab manual and set the Interior-Point-Method as local solver for the Global Search, and the medium scale algorithmic option for the Multistart.

In contrast to the SGDF deterministic starting point strategy, all these derivative free algorithms rely on random starting points or user inputs. Global Search uses a scatter-search mechanism [26] for generating random start points. MultiStart uses uniformly distributed start points within bounds (default 10), or user-supplied starting points. Global Search analyzes starting points and rejects those points that are unlikely to improve the best local minimum found so far. In the contrast, a MultiStart solver runs the local solver from all feasible starting points.

Since these algorithms use random starting points, a single run result could be very poor or excellent. To get a whole picture, we run each solver 26 times (arbitrarily chosen) and use the median number of failures as a measure of its *accuracy measure*. A run fails if its error exceeds 5% in terms of the objective value:

$$\text{error:} = \frac{obj^{*} - obj^{\text{true}}}{\left|obj^{\text{true}}\right|} \tag{22}$$

We use the number of function calls as *efficiency measure*. This is a convention in testing derivative free algorithms since function evaluations are expensive for typical derivative free optimization problems. On the other side, for the purpose of constructing searching directions, Newton and other high order methods spend a significant amount of time on inverting matrix or solving equations; while derivative free algorithms typically don't involve this overhead. Their expense in constructing search directions can also be reasonably measured in number of function evaluations. For SGDF, this expense is on the numerical integration. For other derivative free algorithms, this expense is also on point-wise evaluation based interpolation, pseudo differentiation or simplex derivatives. Again, due to the random starting strategy employed by other derivative free algorithms, we use the median number of function calls of their 26 runs as a robust measure of their efficiency.

We use eight test functions [28]: Brianin, Six-Hump, Goldstein Price, Hartman3, Hartman6, Shekel5, Shekel7, and Shekel10. These functions have known optimal values and optimal solutions, and have been used widely for testing purpose in research papers. For these functions, their dimensions and known optimal values are tabulated in the **Table 1**. We also show the optimal values from SGDF side-by-side with the true optimal value in **Table 1**, which clearly shows that SGDF is capable of achieving high accurate results.

Table 2 shows number of failures for each algorithm as a measure of accuracy. SGDF has no failure, which is also achieved by MS. All other algorithms have difficulty for high dimensional problems. Especially the genetic algorithm and pattern search without polling option exhibit high failure rates.The multi-start strategy (MS), though simple, works very well. SGDF, on the other hand, uses a more complicated strategy. It uses global information of the objective function in the global probing phase, and generates searching directions pointing to region with high objective values on average. This searching heuristics is fundamentally different from the local searching idea. Apparently, this innovative searching heuristics can also be coupled with the simple multistart idea to enhance its success rate. However, multistart inherently increases the computation effort as seen in the next table. Hence how to efficiently combine strength from both methods is a subject of future research.

Table 3 shows the median number of function calls used by each algorithm as a measure of efficiency. It is clear that SGDF uses much less function calls than multistart algorithm. A further calculation shows that SGDF uses 65% less function calls than multi-start algorithm. This is not a surprise since multi-start algorithm runs from ten starting points so as to increase its chance of hitting the global. Global Search also uses multiple starting points, but it drops those unpromising ones sooner. However, Global Search does not simply run local solver from each starting point, it has a more complicated strategies including basin radius estimation, two stage search, and dynamic threshold, etc, which increases its function calls significantly. Pattern Search with additional searching option is more efficient than Pattern

Table 1. Compare the true and SGDF optimal objective values.

Function	D	obj^{true}	obj^{SGDF}	error
Brianin	2	0.398	0.398	0.00%
Four-Hump Camel	2	−1.032	−1.027	0.48%
Goldstein Price	2	3	3.083	2.77%
Hartman3	3	−3.863	−3.827	0.93%
Hartman6	6	−3.322	−3.318	0.12%
Shekel5	5	−10.153	−10.152	0.01%
Shekel7	7	−10.403	−10.359	0.42%
Shekel10	10	−10.536	−10.500	0.34%

Table 2. Median number of failures.

Function	D	SGDF	GA	SA	PS	PS*	GS	MS
Brianin	2	0	0	0	0	0	0	0
Four-Hump Camel	2	0	0	0	1	0	0	0
Goldstein Price	2	0	11	0	0	8	0	0
Hartman3	3	0	1	0	11	7	0	0
Hartman6	6	0	0	0	0	0	0	0
Shekel5	5	0	26	9	23	17	0	0
Shekel7	7	0	26	10	23	19	5	0
Shekel10	10	0	26	13	22	21	1	0

SGDF: sparse grid derivative free; GA: genetic; SA: simulated annealing; PS: pattern search (direct search); PS*: pattern search with polling option on; GS: global search; MS: multi-start.

Table 3. Median number of function calls.

Function	D	SGDF	GA	SA	PS	PS*	GS	MS
Brianin	2	112	1040	1958.5	200	171.5	2331	319
Four-Hump Camel	2	143	1040	1812	200.5	200.5	2310	413
Goldstein Price	2	262	1040	1588	260	210.5	2184	807
Hartman3	3	103	1040	2512	468	316.5	4281	738
Hartman6	6	1069	1100	5785.5	1276	1196	6229.5	1819
Shekel5	5	372	1040	3555.5	583	467	3813	1406.5
Shekel7	7	688	1040	3078.5	581.5	458.5	4386.5	1472.5
Shekel10	10	583	1040	3154	605	435.5	3945	1479

SGDF: sparse grid derivative free; GA: genetic; SA: simulated annealing; PS: pattern search (direct search); PS*: pattern search with polling option on; GS: global search; MS: multi-start.

Search with polling step only. Simulated Annealing and Genetic algorithms apparently use more function calls.

Tables 2 and **3** together show that SGDF is both accurate and efficient. Multi-start is accurate but slightly more expensive. Global Search and Simulated Annealing could be accurate at the expense of more than ten times function calls as SGDF. The Pattern Search with or without searching step is efficient, but could generate a solution far away from optimum if started randomly. Genetic algorithm has reported success for more complicated biological system optimization but does not advantage in this experiment.

It appears that the starting point strategy of SGDF is effective. By using a single starting point, *i.e.*, the global center of gravity, SGDF is able to find solutions very close to a global optimum for eight testing problems. In the contrast, other algorithms use random starting points have more or less fails, except for multi-start algorithm. The multi-start algorithm uses ten starting points in each of the twenty six runs. It is interesting to observe that for this experiment there is always one good starting point out of the ten.

7. Conclusions and Discussion

In this paper, we present a new integration based derivative-free optimization algorithm. The new idea of using integration to generated searching direction has its advantages: we prove that when the integration area is small, the generated direction is always an improving one; we also demonstrate that by simply enlarging the integration area, the algorithm can make use of global information, and generate a searching direction towards area with high objective values on average. We also propose an innovative starting point strategy based on integration. We not only use the modern sparse grid method to implement the integration, but also make contribution in the sparse grid method theory, as we see the need in accurately counting the number of function calls. The new algorithm is also clearly structured in two phases: a global probing phase and a local convergence phase. We equip a new derivative free robust line search method for the algorithm as well. Computational experiments clearly shows that the new heuristic algorithm is accurate and efficient. The result of benchmarking against state-of-art derivative free optimization methods, including Pattern Search, Global Search, Multi-start, Simulated Annealing and Genetic algorithms, is favorable.

The new algorithm is intended to solve practical problems, where derivative information is not available and function evaluation is expensive. For these problems, priority is given to generate a high quality near-optimal solutions within computing budget. On the other hand, we realize that many of these problems also involve integer decision variables, which is not handled by the current algorithm, hence it is a topic for our future research.

REFERENCES

[1] J. D. Pinter, "Global Optimization in Action," Kluwer, Dordrecht, 1996. doi:10.1007/978-1-4757-2502-5

[2] A. J. Booker, J. E. Dennis Jr., P. D. Frank, D. W. Moore and D. B. Serafini, "Managing Surrogate Objectives to Optimize a Helicopter Rotor Design: Further Experiments," *Proceedings of 8th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, 1998.

[3] C. Audet and D. Orban, "Finding Optimal Algorithmic Parameters Using Derivative-Free Optimization," *SIAM Journal on Optimization*, Vol. 17, No. 3, 2006, pp. 642-664.

[4] R. Oeuvary and M. Bierlaire, "A New Derivative-Free Algorithm for the Medical Image Registration Problem," *International Journal of Modelling and Simulation*, Vol. 27, No. 2, 2007, pp. 115-124.

[5] T. Levina, Y. Levin, J. Mcgill and M. Nediak, "Dynamic Pricing with Online Learning and Strategic Consumers: An Application of the Aggregation Algorithm," *Operations Research*, Vol. 57, No. 2, 2009, pp. 327-341.

[6] P. Mugunthan, C. A. Showmaker and R. G. Regis, "Comparison of Function Approximation, Heuristic and Derivative-Based Methods for Automatic Calibration of Computationally Expensive Groundwater Bioremediation Models," *Water Resources*, Vol. 41, 2005.

[7] J. A. Neldder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, Vol. 7, No. 4, 1965, pp. 308-313. doi:10.1093/comjnl/7.4.308

[8] C. Audet and J. E. dennis Jr., "Analysis of Generalized Pattern Searches," *SIAM Journal on Optimization*, Vol. 13, No. 3, 2003, pp. 889-903. doi:10.1137/S1052623400378742

[9] T. G. Kolda, R. M. Lewis and V. Torczon, "Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods," *SIAM Review*, Vol. 45, No. 3, 2003, pp. 385-482. doi:10.1137/S003614450242889

[10] T. A. Winslow, R. J. Trew, P. Gilmore and C. T. Kelley. "Doping Profiles for Optimum Class B Performance of GaAs MESFET Amplifiers," *Proceedings of the IEEE/ Cornell Conference on Advanced Concepts in High Speed Devices and Circuits*, Ithaca, 5-7 August 1991, pp. 188-197.

[11] A. R. Conn, K. Scheinberg and P. L. Toint, "On the Convergence of Derivative-Free Methods for Unconstrained Optimization," *Approximation Theory and Optimization*: *Tributes to M. J. D. Powell*, Cambridge University Press, Cambridge, 1997, pp. 83-108.

[12] M. J. D. Powell, "The NEWUOA Software for Unconstrained Optimization without Derivatives," Technical Report, DAMTP 2004/NA08, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, 2004.

[13] M. Marazzi and J. Nocedal, "Wedge Trust Region Methods for Derivative Free Optimization," *Mathematical Programming*, Vol. 91, No. 2, 2002, pp. 289-305. doi:10.1007/s101070100264

[14] A. R. Conn, K. Scheinberg and L. N. Vicent, "Introduction to Derivative-Free Optimization," *MOS-SIAM Series on Optimization*, SIAM, Philadelphia, 2009. doi:10.1137/1.9780898718768

[15] C. Audet and J. E. Dennis Jr., "Mesh Adaptive Direct Search Algorithms for Constrained Optimization," *SIAM Journal on Optimization*, Vol. 17, No. 1, 2006, pp. 188-217. doi:10.1137/040603371

[16] D. Jones, C. Perttunen and B. Stuckman, "Lipschitzian Optimization without the Lipschitz Constant," *Journal of Optimization Theory Applications*, Vol. 79, No. 1, 1993, pp. 157-181, doi:10.1007/BF00941892

[17] W. Huyer and A. Neumaier, "Global Optimization by Multileve Coordinate Search," *Journal of Global Optimization*, Vol. 14, No. 4, 1999, pp. 331-355. doi:10.1023/A:1008382309369

[18] X. Wang, D. Liang, X. Feng and L. Ye, "A Derivative-Free Optimization Algorithm Based on Conditional Moments," *Journal of Mathematical Analysis and Applications*, Vol. 331, No. 2, 2006, pp. 1337-1360.

[19] G. W. Wasilkowsi and H. Wozniakowski, "Explicit Cost Bounds of Algorithms for Multivariate Tensor Product problems," *Journal of Complexity*, Vol. 11, No. 1, 1995, pp. 1-56. doi:10.1006/jcom.1995.1001

[20] M. S. Bazarra, H. D. Sherali and C. M. Shetty, "Nonlinear Programming: Theory and Algorithms," 3rd Edition, Wiley, John & Sons, Hoboken, 2006.

[21] H.-J. Bungartz and M. Griebel, "Sparse Grids," *Acta Numerica*, Vol. 13, 2004, pp. 147-269.

[22] T. Gerstner and M. Griebel, "Numerical Integration Using Sparse Grid," *Numerical Algorithms*, Vol. 18, 1998, pp. 209-232.

[23] F.-J. Delvos, "D-Variate Boolean Interpolation," *Journal of Approximation Theory*, Vol. 34, No. 2, 1982, pp. 99-114. doi:10.1016/0021-9045(82)90085-5

[24] A. R. Conn, N. I. M. Gould and P. L. Toint, "Trust-Region Methods," *MOS-SIAM Series on Optimization*, SIAM, Philadelphia, 2000. doi:10.1137/1.9780898719857

[25] Z. Ugray, L. Lasdon, J. Plummer, F. Glover, J. Kelly and R. Marti, "Scatter Search and Local NLP Solvers: A Multistart Framework for Global Optimization," *INFORMS Journal on Computing*, Vol. 19, No. 3, 2007, pp. 328-340. doi:10.1287/ijoc.1060.0175

[26] F. Glover, "A Template for Scatter Search and Path Re-

linking," In: J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer and D.Snyers, Eds., *Artificial Intelligence*, Springer, Berlin, Heidelberg, 1998, pp. 13-54.

[27] D. E. Goldberg, "Genetic Algorithms in Search," *Optimization & Machine Learning*, Addison-Wesley, Boston, 1989.

[28] L. C. W. Dixon and G. P. Szegö, "The Global Optimization Problem: An Introduction," *Towards Global Optimisation* 2, North-Holland Pub. Co, Amsterdam, 1978, pp. 1-15.