2017 IEEE 56th Annual Conference on Decision and Control (CDC)
December 12-15, 2017, Melbourne, Australia

Optimization combining derivative-free global exploration with derivative-based local refinement

Shahrouz Ryan Alimo Pooriya Beyhaghi Thomas R. Bewley

Abstract— **This paper proposes a hybrid optimization scheme combining an efficient (and, under the appropriate assumptions, provably globally convergent) derivative-free optimization algorithm (dubbed Δ-DOGS), to globally explore expensive nonconvex functions, with a new derivative-based local optimization algorithm, to maximally accelerate local convergence from promising feasible points discovered in parameter space. The resulting hybrid optimization scheme proceeds iteratively, automatically shifting between (derivative-free) global exploration and (derivative-based) local refinement as appropriate. The new derivative-based local refinement method implemented uses the Voronoi partitioning of all existing datapoints at each iteration to establish a "modified trust-region" around the current best point, within which derivative-based optimization is considered. The resulting algorithm is analyzed, and its global convergence is proven under certain assumptions on the objective function. Finally, the algorithm is applied to nonconvex optimization problems with multiple local minima, and its computational cost compared with that of the original Δ-DOGS algorithm.**

I. Introduction

Consider the optimization of a nonconvex, expensive-to-compute function $f(x)$ with bound constraints,

$$\text{minimize } f(x) \text{ with } \quad x \in B = \{x | a \leq x \leq b\}, \qquad (1)$$

where a and b are two vectors in $\mathbb{R}^n$ such that $a < b$. Solving an optimization problem of the form (1) is difficult and, for general functions, convergence can only be guaranteed if the function evaluation set becomes dense over the entire search domain, B, in the limit of an infinite number of function evaluations [1], [2]. In this paper, for the purpose of derivation of our algorithm, we thus focus our attention on problems in which $f(x)$ is smooth (twice differentiable), and for which the optimization problem considered has a target value f_0; that is, we seek a point $x \in B$ that is a local minimum such that $f(x) \leq f_0$.

There are generally two broad classes of optimization algorithms to solve (1): *derivative-based* methods, which use gradient information to accelerate the search of a local minimum of the objective function, and *derivative-free* methods, which do not use gradient information, but may often be developed in a manner which, under the appropriate assumptions, assures convergence to a global solution to (1) [3], [4], [5].

Derivative-based methods are designed to handle a large number of design parameters, and generally require far fewer function evaluations for local convergence. There are two main approaches for determining the update made at each step of a derivative-based search: those based on *trust-regions*, and those based on *line searches*.

Trust-region methods define, at each iteration, a region in the vicinity of the current point, x_k, within which a model that approximates the objective function is generated and used to calculate the next point, which is restricted to lie within the modified trust-region. Line search methods, in contrast, determine the step length in a chosen search direction via a number of additional function evaluations, in order to identify a point with a reduced function value (see [6], [7]), often coordinated by an *Armijo condition*, which seeks to ensure that the next iterate reduces the function sufficiently relative to the directional derivative of $f(x)$ at x_k in the search direction, or a *Wolfe condition*, which seeks to enforce conditions on ∇f as well as the Armijo condition in order to guarantee that a BFGS update [6] can be safely applied.

Derivative-free methods can, under appropriate assumptions, guarantee convergence to a global optimum, but are generally inefficient computationally, particularly at local refinement, requiring many more function evaluations than derivative-based methods. Response surface methods (RSMs) are the most efficient globally-convergent derivative-free optimization methods available today. RSMs iteratively minimize a search function using an interpolant of existing data points, known as the "surrogate", and a model of the "uncertainty" of this surrogate which goes to zero at the function evaluations themselves. Efficient global optimization (EGO) [8], optimization by radial basis function interpolation in trust-regions (ORBIT) [4], the Surrogate-Management-Framework (SMF) [9], and Delaunay-based derivative-free optimization via global surrogates (Δ-DOGS) [10], [11], [12], are modern examples of RSMs.

The derivative-free scheme upon which the present work is based is Δ-DOGS, which is a generalizable family of computationally-efficient RSMs developed by our group for low-dimensional optimization problems in which the objective function is both nonconvex and expensive to evaluate. There are already a handful of schemes in this family, including schemes designed specifically for simple bound constraints [13], linear constraints [10], [14], and nonconvex constraints [12], [15].

This paper proposes the hybridization of (derivative-free) algorithms in the Δ-DOGS family with a local derivative-based optimization approach in order to significantly accelerate the process of local refinement (for a related discussion, see [5]). The proposed hybrid algorithm inherits the property

Shahrouz Ryan Alimo is with the Dept of MAE, UC San Diego shahrouz.ryan.alimo@gmail.com
Pooriya Beyhaghi is with the Dept of MAE, UC San Diego p.beyhaghi@gmail.com
Thomas R. Bewley is with is with Faculty of the Dept of MAE, UC San Diego bewley@eng.ucsd.edu

978-1-5090-2873-3/17/$31.00 ©2017 IEEE

of global convergence (under the appropriate assumptions) of the particular Δ-DOGS algorithm upon which it is based. In our numerical experiments, the algorithm is found to efficiently handle nonconvex functions with many local minima, and to scale better with dimension than purely derivative-free global optimization approaches.

The paper is structured as follows: Section II briefly reviews the essential ideas of [13], [14], which accelerates a Δ-DOGS search by coordinating it with a Cartesian grid over parameter space that is successively refined as convergence is approached. Section III explains the new hybrid optimization scheme, which combines Δ-DOGS with a derivative-based optimization algorithm, leveraging a modified trust-region approach, for local refinement. Section IV analyzes the new hybrid algorithm's convergence properties, and describes the technical conditions needed to guarantee its convergence to a global or local minimizer. In Section V, the hybrid algorithm is applied to benchmark optimization problems to illustrate its behavior. Conclusions are presented in Section VI.

II. A BRIEF REVIEW OF Δ-DOGS

We now review the essential elements of Δ-DOGS [14], [13]. Note that this paper focuses on variants of these algorithms that leverage target values of f_0; other variants of these algorithms are discussed in [10], [11], [14], [16], and could also be invorporated into this framework.

At each iteration, Δ-DOGS estimates the location in the feasible domain B with, effectively, the highest probability, given the current surrogate and uncertainty models, of having a function value less than or equal to f_0 (that is, which maximizes a probability measure for finding such a function value). The approach is akin to the expected improvement [17] and Bayesian optimization algorithms [18].

Definition 1: Consider $S = \{x_1, x_2, \ldots, x_N\}$ as a set of datapoints in the feasible domain B. The continuous search function $s(x)$ is defined as follows:

$$s(x) = \begin{cases} \frac{p(x)-f_0}{e(x)} & \text{if} \quad p(x) \geq f_0, \\ p(x) - f_0 & \text{otherwise}, \end{cases} \tag{2}$$

where $p(x)$ is some smooth interpolating function such that $p(x_i) = f(x_i), \forall i \in \{1,\ldots,N\}$, and $e(x)$ is an uncertainty function built on the framework of a Delaunay triangulation of existing datapoints; key properties of $e(x)$, as discussed in [10], include it being piecewise quadratic with constant Hessian, $e(x) \geq 0 \ \forall x \in B$, and $e(x_i) = 0 \ \forall i \in \{1,\ldots,N\}$.

Definition 2: The Cartesian grid of level ℓ for the feasible domain $B = \{x | a \leq x \leq b\}$, denoted B_ℓ, is defined such that

$$B_\ell = \left\{ x | x_\iota = a_\iota + \frac{1}{N}(b_\iota - a_\iota) \cdot z_\iota, \quad z_\iota \in \{0, 1, \ldots, 2^\ell\} \right\}$$

A quantizer of a point $x \in B$ onto B_ℓ is a point x_q on the B_ℓ grid with minimum distance to x; note that the quantizer so defined is not necessarily unique. The maximum discretization error is defined as

$$\delta_\ell = \max_{x \in B} \|x - x_q\|. \tag{3}$$

Algorithm 1 Strawman of Δ-DOGS, designed for minimizing $f(x) \in B$ leveraging the target value f_0.

0. Initialize $k = 0$, ℓ, and the initial set of datapoints S_0, and calculate $f(x_i)$ for all $x_i \in S_0$.
1. Calculate or update the interpolating function $p_k(x)$ and the uncertainty function $e_k(x)$ for the points in S_k.
2. Minimize the search function (2) in B to obtain $\hat{x}_k$ as a point with high probability of obtaining the target value.
3. Determine y_k as the quantization of $\hat{x}_k$ on B_{ℓ_k}.
4. If $y_k \notin S_k$, $S_{k+1} = S_k \cup y_k$, and calculate $f(x_k)$; otherwise, refine the mesh by incrementing ℓ_k.
5. Repeat steps 1-4 until a point x is found with $f(x) \leq f_0$.

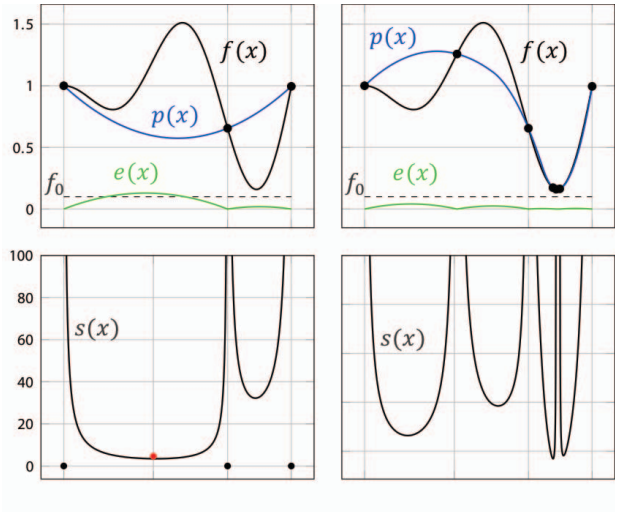


Fig. 1: The essential elements of Algorithm 1, Δ-DOGS, in different iterations for a 1D example. Top subfigures indicate (black) the truth function $f(x)$, (blue) the interpolating surrogate function $p(x)$, (green) the synthetic model of the uncertainty $e(x)$, (dashed) the target value $f_0 = 0.07$, and (black circles) previous datapoints. Bottom subfigures indicate the search function $s(x)$, as defined in (2), and (red circles) the minimizer of $s(x)$, the next datapoint to evaluate.

Illustration of the above concepts can be found in Figure 2 of [13]. The grid B_ℓ has a specific property that is useful in this analysis: if any constraints on B are binding at x, these constraints are also binding at x_q.

Given the above concepts, Algorithm 1 presents a strawman form of the Δ-DOGS algorithm, as illustrated in Figure 1; further details may be found in [13], [14].

Remark 1: At each iteration, Algorithm 1 either adds a new feasible evaluation point, or refines the mesh.

There are two possible termination scenarios for Algorithm 1: either it finds a point x with a function value $f(x) \leq f_0$, or it conducts an infinite number of iterations. In a latter case, it is proved in [13] that there is a limit point amongst the datapoints computed with a function value equal to f_0 if the target value is achievable. This convergence result, related to a limit sequence within the datapoints computed, clearly

represents remarkably faster convergence than the perhaps pessimistic implication of [1], which requires that functions evaluations eventually become dense over the entire feasible domain (in $\mathbb{R}^N$) in the limit of an infinite number of function evaluations.

Though the Δ-DOGS family of schemes is (relative to other derivative-free optimization schemes) quite computationally efficient for the problem of characterizing and globally exploring (via the surrogate) a large range of nonconvex functions, it suffers from the same "curse of dimensionality" that plagues all derivative-free optimization schemes, and scales poorly with the dimension of the problem, n. The hybrid algorithm proposed below mitigates this issue.

III. Accelerating Local Convergence Using a Derivative-Based Method

This section discusses the blending of our globally-convergent derivative-free optimization algorithm, Δ-DOGS, with a local derivative-based optimization approach to accelerate the process of local refinement, and thus to scale better with dimension than purely derivative-free global optimization approaches. The essential idea of the new approach is two-fold: once the Δ-DOGS scheme constructs a reasonably well-sampled surrogate, the best feasible point found thus far is used to initialize a local derivative-based search. Once this derivative-based search identifies a feasible local minimum point, the value and slope of the objective function at this point are used to update the surrogate, and the derivative-free search is resumed, until a new point with an improved objective function value is found, and another derivative-free local refinement is performed, etc.

For the derivative-based component of the above-described hybrid optimization scheme, we will implement a modified trust-region method [6] which iteratively solves the following subproblem:

$$x_k = \text{argmin } q_k(x) \quad \text{subject to} \quad x \in \Omega_k, \tag{4}$$

where Ω_k is a subset of B, S_k is the set of datapoints available at iteration k, and $q_k(x)$ is a local quadratic function constructed around z_k, which is a point in S_k that minimizes $f(x)$, such that $q_k(z_k) = f(z_k)$, $\nabla q_k(z_k) = \nabla f(z_k)$, and $\nabla^2 q_k(z_k) = \nabla^2 f(z_k)$ (or, some approximation thereof).

We now define the modified trust-region Ω_k to be used in the derivative-based component of Algorithm 1. Classical trust-region methods take the trust-region as a sphere around z_k; however, it turns out that this approach does not work particularly well when we combine trust-region-based derivative-free optimization with our global optimization algorithm Δ-DOGS. In this paper, we thus instead define Ω_k as simply the Voronoi cell [19] of z_k in S_k, which is a convex, linearly-constrained region defined as follows.

Definition 3: The *constrained Voronoi cell* around each point $z_k \in S_k$ is consists of all points in B that are closer to z_k than to any other point in $x_j \in S_k$:

$$V(z_k) = \{x \in B \mid \|x - z_k\|_2 \leq \|x - x_j\|_2, \quad \forall x_j \in S_k\}, \tag{5}$$

where $V(z_k)$ represents the constrained Voronoi cell of z_k.

Algorithm 2 The new hybrid optimization algorithm to minimize $f(x)$ in the feasible domain B, leveraging a gradient-based scheme to accelerate local refinement.

0. Initialize $k = 0$, $\ell = \ell_0$, and the initial set of datapoints S_0 (confined to the grid B_ℓ), and calculate $f(x)$ for all points in S_0.
1. Denote z_k as the point in S_k which minimizes $f(x)$. Calculate $\nabla f(z_k)$, calculate or approximate $\nabla^2 f(z_k)$, generate the local quadratic function $q_k(x)$, and solve the constrained quadratic program defined in (6) to obtain x_k.

2a. If (7) is satisfied [i.e., if $q(x_k) < \eta\,(f_0 - f(z_k)) + f(z_k)$], then determine y_k as the quantization of x_k on B_ℓ.

2b. Otherwise [i.e., if (7) is not satisfied], calculate or update the interpolating function $p_k(x)$ and the uncertainty function $e_k(x)$ for the points in S_k, and find the minimum of the search function (2), denoted $\hat{x}_k$, in B. Determine y_k as the quantization of $\hat{x}_k$ on B_ℓ.

3a. If $y_k \notin S_k$, take $S_{k+1} = S_k \cup y_k$, and calculate $f(y_k)$.

3b. Otherwise (i.e., if $y_k \in S_k$), refine the mesh, $\ell \leftarrow \ell + 1$.

4. Repeat from step 1 until convergence.

Taking $\Omega_k = V(z_k)$, the quadratic programming problem in (4) may now be rewritten as

$$x_k = \text{argmin } q_k(x) \quad \text{subject to} \quad x \in V(z_k). \tag{6}$$

We now present, in Algorithm 2, a hybrid optimization algorithm combining Algorithm 1 and the modified-trust-region-based derivative-free optimization method described above. At each iteration, either the quantization of the minimizer (in B) of the search function (2), or the quantization of the solution to the quadratic programming problem (6) (in the modified trust region $V(z_k)$, given by the Voronoi cell surrounding z_k), is added to S_k. For obvious reasons, the first case is called a *global exploration* iteration, and the second case is called a *local refinement* iteration.

The indicator used in Algorithm 2 to select between global exploration and local refinement is the following:

$$q_k(x_k) < \eta \left(f_0 - f(z_k)\right) + f(z_k). \tag{7}$$

If (7) is satisfied, the process of local refinement at this iteration is deemed to be sufficiently promising that it might ultimately lead to a local value of $f(x) \leq f_0$, and thus a (derivative-based) local refinement step is performed; otherwise, a (derivative-free) global exploration step is performed. A single parameter η with $0 < \eta \leq 1$, called the *reduction factor*, is used in this indicator function.

A. Constructing the local quadratic model

We now discuss the construction of the local quadratic function $q_k(x)$. The approach used is based on Quasi-Newton methods, which construct a locally quadratic approximation of the objective function,

$$q_k(x) = f(z_k) + \nabla f(z_k)^T (x - z_k) + \frac{1}{2}(x - z_k)^T H_k (x - z_k), \tag{8}$$

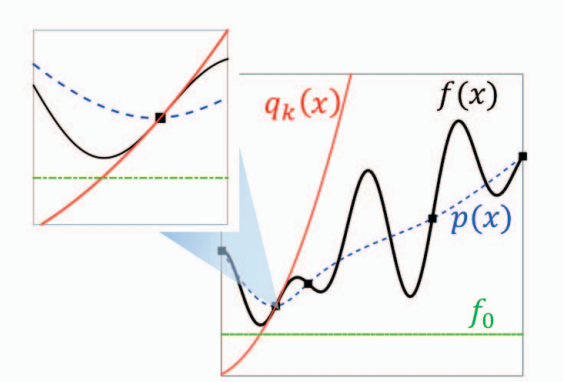


Fig. 2: The quadratic model $q_k(x)$, indicated as red, is a better approximation of the (unknown) objective function $f(x)$ inside the modified trust-region (the Voronoi cell surrounding the best computed point). However, $q_k(x)$ fails to capture the "global trends" of $f(x)$. On the other hand, the surrogate (interpolation) function $p(x)$, indicated as blue, much better summarizes these trends globally, thus motivating our hybrid approach. Note that the target value f_0 is indicated as green.

with the Hessian H_k approximated based on recent gradient computations; this approach can ultimately result in an algorithm with superlinear convergence. In the present work, we use the venerable BFGS method [6] for the construction of H_k. In the implementation of our hybrid approach, the matrix H_k is reinitialized by the identity matrix at any iteration for which the test (7) fails. For each iteration for which the test (7) does not fail, and that a point y_k is obtained such that $f(y_k) \leq f(z_k)$, the matrix H_k is updated via the standard BFGS formula as follows:

$$H_{k+1} = H_k + \begin{cases} \dfrac{\gamma_k^T \gamma_k}{\gamma_k^T d_k} - \dfrac{H_k d_k d_k^T H_k^T}{d_k^T H_k d_k} & \text{if } \gamma_k^T d_k > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{9a}$$

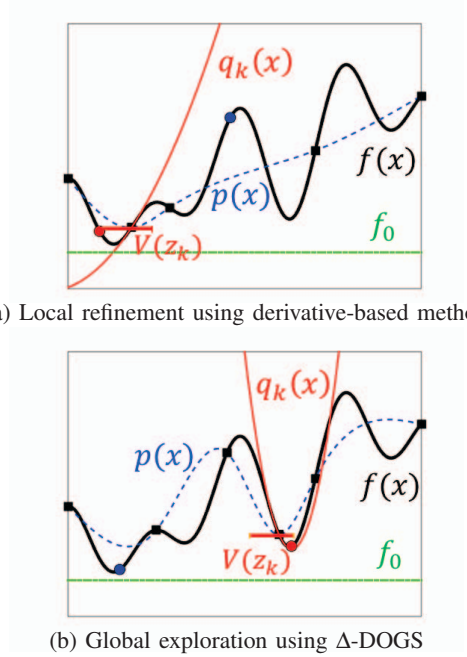
$$d_k = y_k - z_k, \quad \gamma_k = \nabla f(y_k) - \nabla f(z_k). \tag{9b}$$

IV. Analysis

In this section, we analyze the convergence properties of Algorithm 2. Under the appropriate assumptions, we will establish two main properties:

1. If the target value is achievable, the algorithm will either (a) find a feasible point with objective function value less than or equal to f_0 in a finite number of iterations, or (b) if an infinite sequence of points is generated, there will be a limit point amongst the datapoints computed with a function value equal to f_0. This property is called *target achievability*.
2. The algorithm will converge to a KKT point [6] for the objective function $f(x)$. This property is called *local minimum convergence*.

It is established in [13] that Algorithm 1 has the target achievability property; however, Algorithm 1 does not guarantee local minimum convergence. We will establish both properties for Algorithm 2, subject to the following assumptions on the objective function $f(x)$ and the interpolant $p^k(x)$:



(a) Local refinement using derivative-based method

(b) Global exploration using Δ-DOGS

Fig. 3: The two possible scenarios when evaluating (7). In both subfigures, the red circle denotes the x location of the minimum of $q_k(x)$ within $V(z_k)$, as suggested by local refinement, whereas the blue circle denotes the x location of the minimum of $s(x)$ within B, as suggested by global exploration. In (a), a local refinement step will be taken, and in (b) a global exploration step will be taken.

Assumption 1: The interpolating function $p^k(x)$, objective function $f(x)$, and $p^k(x) - f(x)$ are Lipschitz with the same Lipschitz constant $\hat{L}$ in B.

Assumption 2: A constant $\hat{K} > 0$ exists for which

$$\nabla^2\{f(x) - p^k(x)\} + 2\hat{K}I > 0, \quad \forall x \in B \text{ and } k > 0, \tag{10}$$

$$\nabla^2\{p^k(x)\} - 2\hat{K}I < 0, \quad \forall x \in B \text{ and } k > 0, \tag{11}$$

$$\nabla^2\{f(x)\} - 2\hat{K}I < 0, \forall x \in B. \tag{12}$$

Moreover, the gradient of $f(x)$ is Lipschitz with constant K.

Assumption 3: The local quadratic function $q_k(x)$ and its derivative $\nabla q_k(x)$ are Lipschitz with constant $\hat{L}$ inside B.

A. Establishing target achievability of Algorithm 2

By construction, each step of Algorithm 2 is either a local refinement step or a global exploration step. For each mesh refinement iteration of Algorithm 2, there are two possible cases:

(a) Condition (7) is satisfied, or
(b) Condition (7) is not satisfied, but y_k [the quantizer of the minimizer of $s_k(x)$] is located in S_k. See Fig. 3.

It is noted in §5 of [13] that, if an infinite number of steps are taken, then an infinite number of mesh refinement steps are taken; there are thus either an infinite number of mesh refinement steps of the first type above, or an infinite number of mesh refinement steps of the second type above (or, both). Also, by §5 of [13], if there are an infinite number of mesh

refinement iterations that are of the second type above, then Algorithm 2 converges to a point such that $f(x) \le f_0$.

We will now show target achievability when there are an infinite number of mesh refinement steps that satisfy (7).

Theorem 1: If there are an infinite number of iterations k in Algorithm 2 which are mesh refinement and satisfy (7), then

$$\lim_{k\to\infty} f(z_k) \le f_0. \tag{13}$$

Proof: Consider k as an iteration of Algorithm 2, which is a local refinement step and also mesh refining. Then

$$f(z_k) - q(x_k) \ge \eta(f(z_k) - f_0). \tag{14}$$

Since $q(z_k) = f(z_k)$, and $q(x)$ is Lipschitz with constant $\hat{L}$, then

$$f(z_k) - f_0 \le \frac{1}{\eta}\hat{L}\|z_k - x_k\|, \tag{15}$$

On the other hand, step k is mesh refinement. Thus, the quantizer of x_k is in S_k. However, by construction y_k is in the Voronoi cell of z_k. Therefore, z_k is a quantizer of x_k, and

$$f(z_k) - f_0 \le \frac{1}{\eta}\hat{L}\delta_{\ell_k},$$

where δ_{ℓ_k} is the maximum discretization error at iteration k. Since there is an infinite number iterations like k, (13) is shown. ∎

We have thus established that Algorithm 2 will achieve the target value. Moreover, if at one iteration we achieve the target value, then all remaining iterations are local refinement iterations. In the next section, we establish the local minimum convergence of Algorithm 2.

B. Establishing local minimum convergence of Algorithm 2

We first make a few useful definitions.

Definition 4: Define x_k as the solution of the quadratic programming problem (6) at iteration k. There are two possible types of binding constraints at x_k:

a. Constraints on the feasible domain B. These constraints are called *domain-sharing* active constraints.
b. Constraints on the Voronoi cell of z_k. These constraints are called *Voronoi-sharing* active constraints.

Definition 5: Consider $\mathbb{S} = \{V_0, V_1, V_2, \ldots, V_r\}$ as an affinely independent[1] subset of the vertices of a unit n-dimensional hypercube. Then we construct a matrix A as a matrix whose i'th column is $a_i = (V_i - V_0)/\|V_i - V_0\|$. By construction, A is nonsingular. Then, the *hypercube scaling factor* ρ is defined as the inverse of the minimum possible value for $\sigma_{\min}(A)$ (the minimum singular value of A) over all possible subsets of $\mathbb{S}$.

Note that, for each $z \in range(A)$, defined in r-dimensional space, such that $\|z\| = 1$, there is a unique vector $\alpha \in R^r$, s.t.

$$A\alpha = z, \quad \|\alpha\| \le \rho, \quad \sum_{i=1}^{r} |\alpha_i| = \sqrt{r}\rho \le \sqrt{n}\rho.$$

Lemma 1: Consider k as an iteration of Algorithm 2 which is a mesh refinement; then

1. Domain-sharing and Voronoi-sharing constraints are orthogonal.
2. Consider a as the normal vector of a Voronoi-sharing active constraint; then
$$|a^T \nabla f(x)| \le 2\hat{K}\delta_{L_k}, \tag{16}$$
where δ_{L_k} is the maximum discretization error at step k.
3. Consider b as an outward-facing normal vector of a Domain-sharing active constraints; then
$$b^T \nabla f(x) \ge -\hat{K}\,\delta_{L_k}. \tag{17}$$
4. Consider c as a normal vector which is perpendicular to all active constraints at x_k; then
$$|c^T \nabla f(x)| \le \hat{K}\,\delta_{L_k}. \tag{18}$$
5. Consider d as a unit vector which is parallel to the Domain-sharing active constraints at x_k; then
$$|d^T \nabla f(x)| \le (1 + \sqrt{n}\rho)\,\hat{K}\,\delta_{L_k}, \tag{19}$$
where ρ is the scaling factor of the unit hypercube.

Proof: We first show Property 1. Consider H_1 as a boundary of a Voronoi-sharing active constraints, then there is a point $w_k \in S_K$, such that $\|x_k - z_k\| = \|w_k - z_k\|$. By construction, the vector $z_k - w_k$ is orthogonal to H_1. Since step k is a mesh refinement, both z_k and w_k are quantizers of x_k. As a result, according to the construction of the Cartesian grid [13], all domain-sharing active constraints like H_2 are active at both w_k and z_k. Thus, w_k and z_k lie on the boundary of H_2, which establishes Property 1.

To show Property 2, we demonstrate (16) is valid, where a is the normal vector of H_1. According to the mean value theorem, there is a point ξ on the line between z_k and w_k such that

$$\frac{\nabla f(\xi)^T (w_k - z_k)}{\|w_k - z_k\|} = \frac{f(w_k) - f(z_k)}{\|w_k - z_k\|}. \tag{20}$$

Since z_k has the minimum objective value in S_K, then $f(w_k) \ge f(z_k)$. Thus,

$$\frac{\nabla f(\xi)^T (w_k - z_k)}{\|w_k - z_k\|} \ge 0. \tag{21}$$

Moreover, the function $\nabla f(x)$ is Lipschitz; thus,

$$\frac{\nabla f(z_k)^T (w_k - z_k)}{\|w_k - z_k\|} \ge -\hat{K}\|z_k - w_k\| \ge -2\hat{K}\|z_k - x_k\|. \tag{22}$$

On the other hand, x_k is the solution of the quadratic programming problem (6), which is on the constraint H. Moreover, z_k and w_k are infeasible and feasible, respectively, with respect to this constraint. Further, $(w_k - z_k)/\|w_k - z_k\|$ is normal to the boundary of H_1, and goes out of the Voronoi cell. As a result,

$$\frac{\nabla q_k(z_k)^T (w_k - z_k)}{\|w_k - z_k\|} \le 0. \tag{23}$$

[1] s=$\{s_0, s_1, \ldots, s_d\}$ is affinely independent if $\{s_1 - s_0, \ldots, s_d - s_0\}$ are linearly independent.

Since $\nabla q_k(x)$ is Lipschtiz with constant $\hat{K}$, and $\nabla f(z_k) = \nabla q_k(z_k)$, it follows that

$$\frac{\nabla f(z_k)^T (w_k - z_k)}{\|w_k - z_k\|} \leq \hat{K}\|z_k - x_k\|. \tag{24}$$

Since iteration k is mesh decreasing, Property 2 is established.

To show Property 3, consider b as an outward-facing normal vector of a domain-sharing active constraint H_2. Since x_k is the solution of the quadratic programming (6),

$$b^T \nabla q_k(x_k) \leq 0. \tag{25}$$

Since $\nabla q_k(z_k) = \nabla f(z_k)$ and $\nabla q_k(x)$ is Lipschitz with constant K, Property 3 is established.

To show Property 4, since x_k is the solution of the quadratic programming (6), then

$$c^T \nabla q_k(x_k) = 0, \tag{26}$$

Similarly, since $\nabla q_k(z_k) = \nabla f(z_k)$, and $\nabla q_k(x)$ is Lipschitz with constant $\hat{K}$, Property 4 is established.

Finally we consider Property 5. By construction, d can be written as

$$d = d_1 + d_2 \quad \text{where} \quad d_1 = \sum_{i=1}^{r} \alpha_i a_i, \tag{27}$$

where a_i are the normal vectors of the Voronoi-sharing active constraints, and d_2 is a vector which is perpendicular to the domain-sharing active constraints at y_k. Using (16) and (18), and the triangular inequality, we have:

$$|d^T \nabla f(z_k)| \leq \delta_{L_k}[2\hat{K}\sum_{i=1}^{r} |\alpha_i| + \hat{K}\|d_2\|].$$

Furthermore, $\|d\| = 1$, and d_1 and d_2 are orthogonal; thus, $\|d_1\| \leq 1$, $\|d_2\| \leq 1$, and

$$|d^T \nabla f(z_k)| \leq \hat{K}\delta_{L_k}[2\sum_{i=1}^{r} |\alpha_i| + 1].$$

On the other hand, a_i is a vector normal of a boundary of the Voronoi cell of z_k. Therefore, there is a point, $w_i \in S_K$, such that $\|y_k - z_k\| = \|y_k - w_i\|$. Moreover, since iteration k is a mesh refinement, then $\{z_k, w_1, w_2, \ldots, w_r\}$ are distinct quantizers of x_k. As a result, they are located at the vertices of a hypercube. In other words, the a_i are the vectors obtained by connecting one vertex of a uniform hypercube to the other vertices; thus, $\sum_{i=1}^{r} |\alpha_i| \leq \sqrt{n}\rho$, which establishes Property 5. ■

We now prove the local minimum convergence of Algorithm 2.

Theorem 2: Considering $\{k_1, k_2, \ldots,\}$ as the mesh decreasing steps of Algorithm 2, then all limit points of the set $T = \{z_{k_1}, z_{k_2}, \ldots\}$ are KKT points for the optimization problem (1).

Proof: Consider z as a limit point for the set T. Then there is a subset of T like $\{z_{q_1}, z_{q_2}, \ldots\}$, such that

$$\lim_{k\to\infty} z_{q_k} = z. \tag{28}$$

By construction, there is an open ball around z, which does not intersect any boundary of B that does not contain z. Thus, there is a k_0 such that for $k > k_0$, and z_{q_k} could lie only on the boundaries of B that include z. Furthermore, since z_{q_k} is the quantization of x_{q_k}, $A_a(y_{q_k}) \subseteq A_a(z)$, where $A_a(x)$ is the matrix whose rows are the set of active constraints at x in B. As a result, according to Lemma (1), for all $K > \hat{K}$,

$$|p^T \nabla f(z_{q_k})| \leq (1 + \sqrt{n}\rho) K \delta_{L_{q_k}}, \forall p \in null(A_a) \tag{29}$$

$$p^T \nabla f(z_{q_k}) \geq -(1 + \sqrt{n}\rho) K \delta_{L_{q_k}}, \forall p \in row(A_a) \tag{30}$$

Since $\delta_{L_{q_k}}$ converges to zero, z is a KKT point [6]. ■

V. Results

In this section, we compare the performance of (a) the original Algorithm 1, (b) Algorithm 2 with steepest descent applied for local refinement, (c) Algorithm 2 with the BFGS formula applied for local refinement, and (d) the active-set, derivative-based, method of [6]. The function considered is the n-dimensional Styblinski Tang function, which is a benchmark test for global optimization:

$$f(x) = \sum_{i=1}^{n} \frac{x_i^4 - 16x_i^2 + 5x_i}{2} - 39.16616n, \tag{31}$$
$$\text{where} \quad L = \{x | -5 \leq x_i \leq 5\}.$$

An initial grid level of $\ell_0 = 3$ is considered, and the algorithm continues until the grid level of $\ell = 8$ is terminated. Note that the optimizations are terminated when $\|x_k - x_j\|_2 \leq 0.005$ for all $x_j \in S_k$, which leads to a comparable order of accuracy for methods (a), (b), and (c) (i.e. the maximum discretization error level $\ell = 8$ is close to 0.005). The initial datapoints in S_E^0 are constructed with $n+1$ points as follows:

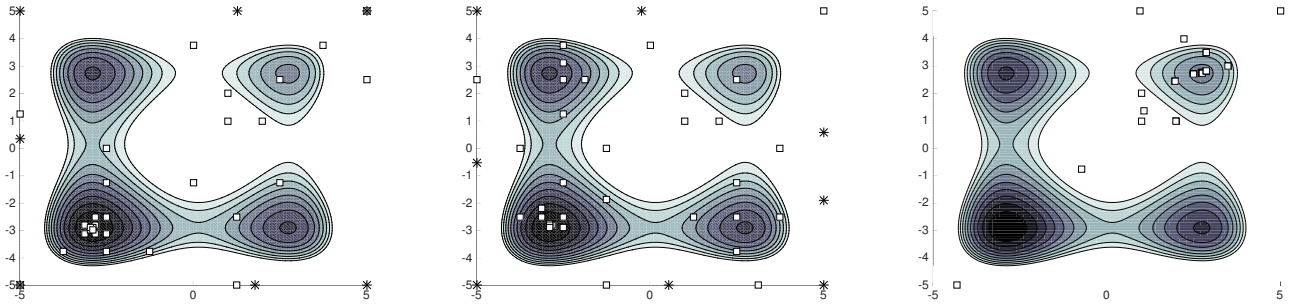
$$S_E^0 = \left\{ x_0, x_0 + \frac{b_i - a_i}{2^{\ell_0}} e_i, \forall i \in \{1, 2, \ldots, n\} \right\}. \tag{32}$$

For each i, e_i is the i^{th} main coordinate direction, and x_0 is an initial point on the grid of level ℓ_0. In this section, we consider two different points of x_0 for the initialization of Algorithms 1 and 2, as shown in Figs. 4 and 6.

Fig. 4 illustrates the position of the datapoints that are used during the optimization process for $n = 2$. With initial points $(x_0 = 0.55, x_0 + 0.2e_i)$, which are far from all local minima, Algorithm 2 focuses on global exploration; as a result, the number of function evaluations required for convergence is similar to that required by Algorithm 1.

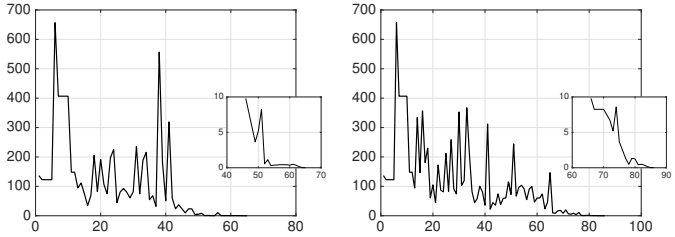
Conversely, with an initial point that lies close to a local minimum, Algorithm 2 performs a much more efficient local refinement than Algorithm 1, resulting in much faster convergence. Table I reports noticeable differences that indicate significant advantages for using Algorithm 2 in higher dimensional problems.

As described in §3.B, the local refinement of Algorithm 2 can incorporate either gradient information or an approximation of the Hessian using the BFGS update formula. Table I demonstrates, as expected, that using the Hessian



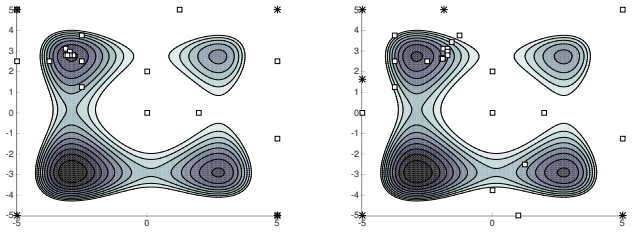
(a) Algorithm 2 w/ BFGS, $\eta = 0.5$. (b) Algorithm 1. (c) Derivative-based method [7].

Fig. 4: Optimization in the case of exact $f_0 = f(x^*) = 0$, with $n = 2$. The black stars are support points (see [13]) that are not actually evaluated, and are only used to regularize the Deluanay triangulation constructed. The white squares are the points at which function evaluations are performed. The darker contours indicate lower values of the objective function whereas the lighter contours indicate higher values. Note that case (c) prematurely converges to a local minimum after 23 iterations.



(a) Algorithm 2 w/ BFGS, $\eta = 0.8$. (b) Algorithm 1.

Fig. 5: Exact $f_0 = f(x^*) = 0$, with $n = 4$.



(a) Algorithm 2 w/ BFGS, $\eta = 0.8$. (b) Algorithm 1.

Fig. 6: Target achievability with $f_0 = 20 > f(x^*) = 0$. In this situation, Algorithm 2 can guarantee the convergence to a local solution; however, Algorithm 1 does not guarantee to find a local solution. See Figure 4 for description of plots.

TABLE I: Algorithm 2 with $\eta = 0.8$, and with steepest descent and BFGS for local refinement, vs. Algorithm. 1. Results averaged over 5 different initial values in each case.

Average # fun. eval. / Dimension	$n = 2$	$n = 3$	$n = 4$
Algorithm 1 Δ-DOGS [13]	22.5	49	98.5
Algorithm 2 with BFGS	25	38	77.8
Algorithm 2 with steepest descent	27.2	61.2	59.4

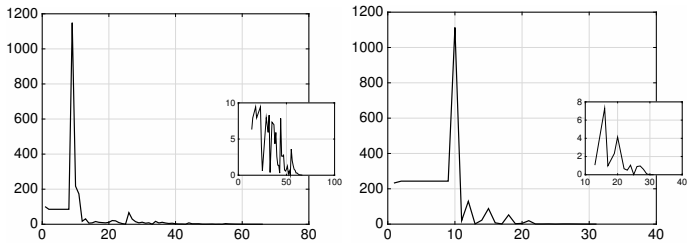
approximation generally has a superior convergence rate as compared with using steepest descent. Additionally, it is observed that the accuracy of the solution is significantly improved for a fixed number of function evaluations when the BFGS update formula is used. As expected, in the case of Hessian approximation, the grid B_{ℓ_0} is refined faster than when using gradient information only.

Algorithm 2 with gradient descent in some situations got stuck at a local solution, and performed many unnecessary function evaluations before starting to explore more globally. Due to this issue in some specific situations Algorithm 2 with gradient descent becomes more computationally expensive than Algorithm 1 and Algorithm 2 with BFGS.

In the case that the estimated solution, f_0, is greater than the global solution, $f(x^*)$, we see another significant advantage of Algorithm 2 over Algorithm 1. Algorithm 1 persists in using global search to find f_0, and stops without convergence using local refinement; thus it does not guarantee to even find a local solution when $f_0 > f(x^*)$. On the other hand, Algorithm 2 continues its local refinement until it converges to a KKT point. This is illustrated in Fig. 6. Finally, the derivative-based method converges to a local solution Fig. $6(c)$.

The cost of computing a Delaunay triangulation grows exponentially as the dimension of the problem grows. Using Algorithm 2 with a good initial guess, the new algorithm can converge to the global solution with a reasonable number of function evaluations even up to $n = 8$, as shown in Fig. 7.

The parameter η specifies the trade-off between global exploration and local refinement. It is desirable to decrease η as the dimension of the problem is increased to emphasize local refinement, as derivative-free global exploration becomes computationally expensive in high-dimensional problems. In the case of a low-dimensional problem ($n < 6$), the performance of the algorithm is not unduly sensitive to the choice of η, and in these cases we have taken $\eta = 0.8$ in the simulations reported here . As the dimensionality of the problems was increased, we generally found that reducing the value of η was beneficial, in order to focus more heavily on local refinement. The optimal value of η for any given problem is likely closely related to some measure



(a) $n = 7$. (b) $n = 8$.

Fig. 7: Algorithm 2 with BFGS applied to example problems with different initial points. (a) $n = 7$, $\eta = 0.4$, 106 local refinement steps, 11 global exploration steps, and 7 times mesh refinement steps, with 8 initial points $\{x_{0,i} = 0.13, x_{0,i} + 0.1\,e_i\}$. (b) $n = 8$, $\eta = 0.1$, 36 local refinement steps, 11 global exploration steps, and 8 mesh refinement steps, with 9 initial points $\{x_{0,i} = 0.4, x_{0,i} + 0.1\,e_i\}$. Note that optimizations in these higher dimensions was simply not possible using Algorithm 1, due to the high computational cost of computing Delaunay triangulations in these dimensions.

of the curvature of the objective function over the feasibility domain. Unfortunately, this quantity would almost never be known in advance, and we therefore suggest tuning it based on a minor amount of trial and error on related problems.

VI. Conclusions

This paper introduces a modification to the Delaunay-based derivative-free optimization algorithm scheme Δ-DOGS, as proposed in [13], [14], incorporating gradient information to accelerate local refinement. The new scheme, Algorithm 2, has three main modifications as compared with the original Δ-DOGS algorithm:

- A criterion, (7), for the anticipated reduction due to a local refinement step is introduced to decide between taking a derivative-based local refinement step or a derivative-free global exploration step at each iteration. This criterion has an adjustable parameter η; values in the range $0.5 \leq \eta \leq 0.8$ were found to be effective.
- A new derivative-based local optimization method is used leveraging a modified trust region approach based on the Voronoi cell of the available datapoints constrained to the (bound) feasible domain. To guarantee convergence, all of the datapoints computed are coordinated by a grid, with this grid being successively refined as the optimization algorithm proceeds.
- To accelerate the convergence of local refinement scheme and the hybrid method that uses it, Algorithm 2, the Hessian of objective function is approximated via the usual BFGS formula.

Proof of global convergence of the new scheme, under the appropriate assumptions, is established. Further, in the numerical experiments we have performed thus far, Algorithm 2 is found to significantly accelerate local convergence, to handle efficiently nonconvex functions with many local minima, and to scale better with dimension than purely derivative-free global optimization approaches.

In future work, this framework will be applied to various additional benchmark problems as well as application-based problems, e.g. [18], [20]. In many online application-based optimization problems, the desire is to find as good a solution as possible within a specific time horizon. Certain modifications of η and f_0 as the time horizon runs out might well be warranted in such situations.

Acknowledgement

The authors gratefully acknowledge funding from AFOSR FA 9550-12-1-0046 in support of this work.

References

[1] Torn, A., & Zilinskas, A. (1989). Global optimization. Springer-Verlag New York, Inc..

[2] Stephens, C. P., & Baritompa, W. (1998). *Global optimization requires global information*. Journal of Optimization Theory and Applications, 96(3), 575-588.

[3] Conn, A. R., & Le Digabel, S. (2013). Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. Optimization Methods and Software, 28(1), 139-158.

[4] Wild, S. M., Regis, R. G., & Shoemaker, C. A. (2008). ORBIT: Optimization by radial basis function interpolation in trust-regions. SIAM Journal on Scientific Computing, 30(6), 3197-3219.

[5] Schonlau, M., Welch, W. J., & Jones, D. R. (1998). Global versus local search in constrained optimization of computer models. Lecture Notes-Monograph Series, 11-25.

[6] Nocedal. J & Wright. S.J.: Numerical Optimization, Springer (2006)

[7] Gill, P. E., & Wong, E. (2015). Methods for convex and general quadratic programming. Mathematical Programming Computation, 7(1), 71-112.

[8] Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. Journal of Global optimization, 13(4), 455-492.

[9] Booker, A. J., Dennis Jr, J. E., Frank, P. D., Serafini, D. B., Torczon, V., & Trosset, M. W. (1999). A rigorous framework for optimization of expensive functions by surrogates. Structural optimization, 17(1), 1-13.

[10] Beyhaghi, P., Cavaglieri, D., & Bewley, T. (2015). Delaunay-based derivative-free optimization via global surrogates, part I: linear constraints. Journal of Global Optimization.

[11] Beyhaghi, P., & Bewley, T. (2016). Delaunay-based derivative-free optimization via global surrogates, part II:convex constraints. Journal of Global Optimization.

[12] Alimo, S. R., Beyhaghi, P., & Bewley. T. : Delaunay-based Derivative-free Optimization via Global Surrogates, Part III: nonconvex constraints. Journal of Global Optimization. Under review.

[13] Beyhaghi. P., & Bewley. T.: Implementation of Cartesian grids to accelerate Delaunay-based Optimization. Journal of Global Optimization. Under review.

[14] Alimo, S. R., Beyhaghi, P., & Bewley, T. : *Implementation of dense lattices to accelerate Delaunay-based optimization.* Optimization Methods and Software Journal, Under review.

[15] Alimo. S. R., Cavaglieri, D., Beyhaghi, P., & Bewley, T. R. : *Discovery of an IMEXRK time integration scheme via Delaunay-based derivative-free global optimization*, J. Opt. & Eng. Under preparation.

[16] Alimohammadi, S., Beyhaghi, P., Meneghello, G., & Bewley, T. (2017): *Delaunay-based optimization in CFD leveraging multivariate adaptive polyharmonic splines (MAPS).* In 58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference (p. 0129).

[17] Gramacy, R. B., & Lee, H. K. H. (2008). Bayesian treed Gaussian process models with an application to computer modeling. Journal of the American Statistical Association, 103(483), 1119-1130.

[18] Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In Advances in neural information processing systems (pp. 2951-2959).

[19] Conway, J. H. & Sloane, N.J.A. (1998) Sphere packings, lattices and groups. Grundlehren der mathematischen Wissenschaften.

[20] Alimohammadi, S., & He, D. (2016, July). *Multi-stage algorithm for uncertainty analysis of solar power forecasting*. In Power and Energy Society General Meeting (PESGM), 2016 (pp. 1-5). IEEE.