Computational Optimization and Applications (2018) 71:307–329
https://doi.org/10.1007/s10589-018-0020-4

A progressive barrier derivative-free trust-region algorithm for constrained optimization

Charles Audet[1] · Andrew R. Conn[2] · Sébastien Le Digabel[1] · Mathilde Peyrega[1]

Received: 28 June 2016 / Published online: 6 July 2018

Abstract
We study derivative-free constrained optimization problems and propose a trust-region method that builds linear or quadratic models around the best feasible and around the best infeasible solutions found so far. These models are optimized within a trust region, and the progressive barrier methodology handles the constraints by progressively pushing the infeasible solutions toward the feasible domain. Computational experiments on 40 smooth constrained problems indicate that the proposed method is competitive with COBYLA, and experiments on two nonsmooth multidisciplinary optimization problems from mechanical engineering show that it can be competitive with the NOMAD software.

Keywords Derivative-free optimization · Trust-region algorithms · Progressive barrier

Mathematics Subject Classification 90C30 · 90C56

✉ Mathilde Peyrega
mathilde.peyrega@polymtl.ca

Charles Audet
https://www.gerad.ca/Charles.Audet

Andrew R. Conn
arconn@us.ibm.com

Sébastien Le Digabel
https://www.gerad.ca/Sebastien.Le.Digabel

1 GERAD and Département de mathématiques et génie industriel, École Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville, Montréal, QC H3C 3A7, Canada

2 IBM Business Analytics and Mathematical Sciences, IBM T J Watson Research Center, P. O. Box 218, Yorktown Heights, NY 10598, USA

1 Introduction

This work targets inequality constrained optimization problems by combining ideas from unconstrained derivative-free trust-region algorithms (DFTR) [39] with the progressive barrier (PB) approach [8] to handle constraints. We consider the following optimization problem:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{subject to} \quad & c(x) \leq 0 \\ & l \leq x \leq u \end{aligned} \tag{1}$$

where $f \colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a single-valued objective function, $c \colon \mathbb{R}^n \to (\mathbb{R} \cup \{+\infty\})^m$ corresponds to the vector of constraints, $l, u \in (\mathbb{R} \cup \{\pm\infty\})^n$ are bounds and $n, m \in \mathbb{N}$. The functions f and c are typically the outputs of a simulation, or a blackbox, and are called the true functions, while Problem (1) is called the true problem.

The following terminology from the taxonomy presented in [32] is used. The constraints of Problem (1) are assumed to be *quantifiable*, *relaxable*, *simulation*, and *known*. These assumptions have the following meaning: a relaxable constraint can be violated without causing issues in the evaluation of the objective and the other constraint functions, whereas the violation of any unrelaxable constraint makes the other outputs non exploitable. In practice, it means that, for any algorithm, infeasible points may be considered as iterates as long as the proposed solution is feasible. Quantifiable means that the function c returns a vector of values and it is possible to measure, from an infeasible point, the distance to feasibility. Simulation means that the analytical expressions of the constraints are not available, but rather given by a simulation, and finally, there are no hidden constraints, which are constraints not known by the user.

Derivative-free methods from the literature may be classified into direct-search and model-based algorithms. At each iteration of a direct-search method, decisions for the next iterations are based only on the comparison of the newly evaluated points with the best solution so far. In model-based methods, local models are built around the current iterate. Both approaches have certain advantages. For example direct-search methods are simpler and can be more easily parallelized but on the other hand model based-methods try and account for the shape of the function more directly. Given a very badly behaved function we would use a direct-search method. If the function can be adequately approximated by a smooth function we would prefer a model-based approach unless it is essential to exploit some parallel architecture.

Many derivative-free algorithms and their applications in numerous fields are discussed in the textbooks [10,21] and in the surveys [3,22].

The treatment of nonlinear constraints remains a real challenge. Of course unconstrained methods can always be applied inside frameworks such as exact penalty methods [35], Lagrangian methods [15], or sequential penalty derivative-free methods [36]. Our approach is rather a direct treatment for general constraints, which only a few algorithms offer.

In model-based algorithms, the first algorithm proposed to handle general constraints without the use of their derivative is COBYLA designed by Powell and presented in [39]. It is a derivative-free trust-region algorithm and the constraints are treated in the subproblem. Linear models are built from the vertices of simplexes. In [16] a DFTR algorithm is proposed to treat problems without the use of the objective function derivatives but with the gradient of nonlinear constraints. In [46] a DFTR algorithm combines an augmented Lagrangian method with a filter technique to solve specific optimization problems presenting a separable structure. In [41] and [42], Sampaio and Toint propose to use the trust-funnel method of Gould and Toint [27] for problems with general constraints. In [45], Tröltzsch adapts SQP algorithms to general equalities in a derivative-free algorithm. In [13] general inequality constrained problems are solved by a DFTR algorithm called NOWPAC. At each iteration, the strict feasibility of the trial point is ensured, thanks to an interior path provided by a quadratic offset of the constraints. NOWPAC requires feasible initial points. Finally, in [2] and [24] DFTR algorithms using an inexact restoration method are proposed with respectively a penalty-like merit function and a filter. Algorithm treating linear constraints are proposed in [28] and [40] for model-based methods.

In the direct-search class of methods, the extreme barrier [5] treats all types of constraints by rejecting infeasible points. This is achieved by setting the objective function value to infinity at infeasible trial points. Filter methods [25] are adapted in [1], [6] and [23] to direct-search methods to treat nonlinear and quantifiable constraints. Filter methods do not combine the objective and the constraints into a single merit function as penalty-based methods, but instead they aggregate all constraints into a single constraint violation function and consider trade-offs between minimizing the objective function versus reducing the constraint violation. Kolda, Lewis and Torczon [29] present an algorithm for problems with general equality constraints where linear equalities are treated explicitly, while nonlinear constraints are handled by an augmented Lagrangian method adapted from [17] to a direct-search algorithm. Derivatives of nonlinear constraints are not used. In 2009, the progressive barrier (PB) for inequalities was proposed and specialized to the mesh adaptive direct search algorithm (MADS) [8]. The PB treats inequality constraints by aggregating constraints violations into a single function and considers trade-offs between objective function quality and feasibility. The name of the method originates from an upper bound, a threshold on the constraint violation, that is progressively reduced to force the iterates towards the feasible region. There is no need of derivatives and no penalty function. In 2010, the progressive-to-extreme barrier, [9], combines both progressive and extreme barrier techniques. Specific treatment for linear equalities or inequalities are also proposed in [12,30,33,34] for direct-search methods.

The goal of this work is to adapt the constraints handling PB technique to the DFTR framework and to demonstrate the potential of this hybridization though computational experiments. The document is organized as follows. Section 2 gives a high level description of a generic DFTR algorithm for unconstrained optimization, followed by the main PB components necessary to handle constraints. A basic framework for DFTR algorithms and a short description of the techniques used to build and improve the models are given. The PB is presented as a process to treat the constraints that allows infeasible iterates. Section 3 introduces a new algorithm combining the DFTR framework and the PB to solve Problem (1). Computational experiments are conducted in Sect. 4 using a Python implementation on two sets of problems. We first consider a set of 40 analytical problems from the CUTEst collection [26], and then two realistic multidisciplinary design optimization (MDO) problems from the mechanical engineering literature [43,44]. Our code is shown to be competitive with COBYLA for DFO problems, and competitive with NOMAD [31] for BBO problems. We conclude with a discussion and future work in Sect. 5.

2 Derivative-free trust-region and progressive barrier

2.1 Trust region notations and definitions

The DFTR algorithms for unconstrained optimization belong to the class of model-based algorithms, but also to the class of trust-region algorithms. The unconstrained DFTR algorithm targets the following derivative-free optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x). \tag{2}$$

Similarly to the case with derivatives, such as in [47], the trust-region algorithm works efficiently provided that one can trust the model in a neighborhood of the current iterate. This neighborhood is called the trust region. At each iteration a (relatively) local model of the objective function is built and then minimized, [21]. In DFTR methods, the gradient of the objective function is unavailable to build the models, although the convergence analysis assumes some smoothness properties of the objective function and some properties of the model. Below, we define the trust region, and some notions for the models and the geometry of the sample set used to build these models. The main idea, as in classical trust-region methods with derivatives, is to solve the subproblems defined with model functions that are easier to minimize, and to perform this minimization within the trust region.

The following description is inspired by [20], with small modifications in the management of the criticality step. As in standard trust-region methods with derivatives, x^k denotes the current iterate at iteration k, and the trust region is the closed ball $B(x^k; \Delta^k) = \{x \in \mathbb{R}^n \ : \ \|x - x^k\| \leq \Delta^k\}$. The norms used in practice include Euclidian and infinite norms. The size of the trust region, within which the model is optimized, is called the trust region radius and is denoted by the scalar $\Delta^k > 0$.

In the theory developed in [21], at each iteration k the model is built through interpolation or regression from a finite set $\mathcal{Y}^k(x^k) \subset \mathbb{R}^n$ of sample points for which the objective function has been evaluated. Hence derivatives are not used to construct the model. The model of the true function f at iteration k is denoted by $\tilde{f}^k$.

The convergence analysis relies on some assumption on the quality of the models. It is convenient to assume that the functions satisfy the following definition proposed in [21] and slightly reformulated in [14]. Indeed, the definition ensures first-order convergence results similar to those of classical trust-region methods because the model has Taylor-like first-order behaviour. This definition applies in particular to continuously differentiable functions with Lipschitz continuous gradient on an appropriate open domain (see [21, chap. 6]).

Definition 2.1 Let $\tilde{f}$ be a model of $f \in \mathcal{C}^1$ at $x \in \mathbb{R}^n$, and consider a radius $\Delta > 0$. Suppose that there exists a positive constant κ such that for all $y \in B(x;\Delta)$ the model $\tilde{f}$ satisfies :

$$\begin{aligned} \|\nabla f(y) - \nabla \tilde{f}(y)\| &\leq \kappa\Delta, \\ |f(y) - \tilde{f}(y)| &\leq \kappa\Delta^2. \end{aligned}$$

Then the model $\tilde{f}$ is said to be a κ-fully-linear model of f on $B(x;\Delta)$.

These properties indicate that the model $\tilde{f}$ and its gradient are close to the function f and its gradient ∇f as long as the trust region radius is small enough. Similarities with Taylor bounds when the derivative are available are clear.

The second-order convergence results require the models to satisfy the following definition:

Definition 2.2 Let $\tilde{f}$ be a model of $f \in \mathcal{C}^2$ at $x \in \mathbb{R}^n$, and consider the radius $\Delta > 0$. Suppose that there exists a positive constant κ such that for all $y \in B(x;\Delta)$ the model $\tilde{f}$ satisfies:

$$\begin{aligned} \|\nabla^2 f(y) - \nabla \tilde{f}(y)\| &\leq \kappa\Delta, \\ \|\nabla f(y) - \nabla \tilde{f}(y)\| &\leq \kappa(\Delta)^2, \\ |f(y) - \tilde{f}(y)| &\leq \kappa\Delta^3. \end{aligned}$$

Then the model $\tilde{f}$ is said to be a κ-fully-quadratic model of f on $B(x;\Delta)$.

Once again similarities with Taylor-like second-order bounds can be observed. Assuming that the model satisfies stronger properties leads to stronger convergence results (e.g., see [13] and their definition of p-reduced fully-linear).

In [21, Chap. 6], different algorithms are detailed to construct and maintain fully-linear and fully-quadratic models. They are based on the notion of well-poisedness introduced in [21, chap. 3]. A model is said to be certifiably fully-linear (respectively certifiably fully-quadratic) when the sample set satisfies well-poisedness properties. These geometric properties on sample sets are sufficient conditions to ensure fully-linear or fully-quadratic models, and convergence is guaranteed, essentially because in the former case we know that the model improves as the trust region radius is decreased so the trust region management alone ensures that the radius stays bounded away from zero without taking any special precautions. We remark that well-poised sample sets can be determined in a finite number of steps.

2.2 The unconstrained DFTR algorithm

A basic framework for unconstrained DFTR algorithms is summarized in this section. The model is built using interpolation techniques rather than Taylor approximations, which implies some distinctions between the DFTR algorithm and a classical trust-region algorithm with derivatives.

There are different ways to define a DFTR algorithm, in particular different options to choose the sample sets and build the models [42]. To simplify and to present the main steps of the algorithm, at each iteration k, we build certifiably κ-fully-linear models for some fixed $\kappa > 0$.

At each iteration the well-poisedness of the sample set is checked and modified if necessary. In our algorithm the sample set $\mathcal{Y}^k(x^k)$ is built by taking exactly $(n+1)$ points for linear models and $\frac{(n+1)(n+2)}{2}$ for quadratic models in a ball of radius $2\Delta^k$ around x^k.

Only interpolation techniques are used, no regression is involved. If there is an insufficient number of points then additional points are randomly sampled and the geometry improvement algorithm is called, before evaluating the true function values. If there are too many points, the most recent points are chosen.

Algorithm 1 DFTR for unconstrained optimization: Iteration k.

Step 1 - Model construction
Build a quadratic model $\tilde{f}^k$ from $\mathcal{Y}^k(x^k)$ that approximates the objective function f in $B(x^k;\Delta^k)$.

Step 2 - Subproblem
Optimize the subproblem on the trust region:

$$\tilde{x}^k \in \underset{x\in B(x^k;\Delta^k)}{\text{argmin}} \ \tilde{f}^k(x). \tag{3}$$

The step $\tilde{s}^k = \tilde{x}^k - x^k \in B(0;\Delta^k)$ must achieve a fraction of the Cauchy step s_C^k.

Step 3 - Step calculation
Evaluate f at $\tilde{x}^k$ and compute the ratio

$$\rho_f^k = \frac{f(x^k) - f(\tilde{x}^k)}{\tilde{f}^k(x^k) - \tilde{f}^k(\tilde{x}^k)}.$$

Step 4 - Trust region radius update

- If $\rho_f^k \geq \eta_1$, then set $x^{k+1} = \tilde{x}^k$ and
$$\Delta^{k+1} = \begin{cases} \gamma_{dec}\Delta^k & \text{if } \Delta^k > \mu\|g^k\|, \\ \min\{\gamma_{inc}\Delta^k, \Delta_{max}\} & \text{if } \Delta^k \leq \mu\|g^k\|. \end{cases}$$
- If $\eta_0 \leq \rho_f^k < \eta_1$, then set $x^{k+1} = \tilde{x}^k$ and
$$\Delta^{k+1} = \begin{cases} \gamma_{dec}\Delta^k & \text{if } \Delta^k > \mu\|g^k\|, \\ \Delta^k & \text{if } \Delta^k \leq \mu\|g^k\|. \end{cases}$$
- If $\rho_f^k < \eta_0$, then set $x^{k+1} = x^k$ and $\Delta^{k+1} = \gamma_{dec}\Delta^k$.

The outline of this method is presented in Algorithm 1. Additional algorithmic parameters are defined by: η_0, η_1, γ_{dec}, γ_{inc}, μ with $0 \leq \eta_0 \leq \eta_1 < 1$ (and $\eta_1 \neq 0$), $0 < \gamma_{dec} < 1 < \gamma_{inc}$, $\mu > 0$. The parameter η_1 represents a threshold to decide if the ratio comparing true and predicted improvements of the objective function is sufficiently high. The parameter η_0 is non-negative and can be chosen equal to zero. It is introduced, among other reasons, to allow simple decrease rather than sufficient decrease. The constants γ_{dec} and γ_{inc} are multiplication factors to increase and decrease the trust region radius Δ^k. The parameter μ is a constant used to trigger the decreases of the trust region radius when the gradient becomes small, to ensure that for small values of Δ^k the difference between the true gradient (or some gradient of a nearby function) is appropriately approximated by the model gradient. A starting point x^0 must also be provided. In Step 2, the Cauchy step s_C^k is the minimizer of the model in the direction of the gradient. The notation g^k refers to the gradient of the model function $\tilde{f}^k$ and it is the direction to the Cauchy point that drives first-order descent (corresponding to the steepest descent direction in methods with derivatives). Step 4 is directly inspired from the simple trust-region algorithm presented in [14].

As detailed in [20], which presents a general convergence analysis for DFTR algorithms, when the step is successful, if the model gradient is not small and the ratio for the trust region management is large enough, we increase Δ^k regardless of any other condition.

If the trust region radius is large compared to the norm of the model gradient, then the characteristics of a fully-linear or fully-quadratic model are not helpful in the sense that, for example, a small gradient for the model tells us little about the size of the true gradient. It is for this reason that we may need to reduce the trust region size in a manner that relates to the model gradient size to ensure that we can conclude that the stationarity condition is meaningful. Under additional assumptions on f, and by replacing the original criticality step by the one described in [14], first and second order global convergence can be proved [21]. The models are fully-linear for first order analysis and fully-quadratic for second order analysis. Step 4 in Algorithm 1 ensures that Δ^k converges to zero and replaces both Step 5 and the criticality step in the algorithm proposed in [21, chap. 10]. Indeed, if there are infinitely many iterations in which the trust region radius is not decreased, then there are infinitely many iterations in which the value of f decreases by a minimal value, which is impossible if f is assumed to be bounded from below. A natural stopping criteria is then based on the value of Δ^k.

As the trust region radius converges to zero, it means with the definition of fully-linear and fully-quadratic models that the models become sufficiently close to the true function on the trust region. As we can prove that the model gradient converges to zero because a fraction of a Cauchy step is achieved at each iteration, the true gradient also converges to zero. In other words, convergence analysis in classical trust-region algorithm with derivatives can be transferred to the derivative-free trust-region algorithm under an appropriate assumption. The most important difference is that in classical trust-region algorithm with sufficient decrease, the trust region radius is bounded away from zero [18, Lem 6.5.4] whereas it converges to zero in the DFO context. However, and contrary to [48], in the derivative case one can guarantee convergence with simple decrease if the trust region radius Δ^k goes to zero, but this may not be a good idea in practice.

An important feature of Algorithm 1 is that it is not required to impose fully-linear (or fully-quadratic) models at every iteration, but only at the ones where the trust region radius is decreased or if the model gradient is (relatively) small compared with the trust region radius, and we are hoping this means that we are close to stationarity, so we have to ensure that the model gradient adequately represents the true gradient.

In our presentation, the algorithm certifying the well-poisedness of the sample set is called at the beginning of each iteration. Thus the models are always certifiably fully-linear or fully-quadratic. If the well-poisedness is not guaranteed, an algorithm to improve the geometry of the sample set is called. Hence, when the ratio ρ_f^k of Step 3 is smaller than η_0, and when in addition the gradient of the model is large in comparison with the trust region radius Δ^k, we can reduce the trust region radius to improve the accuracy of the model on a smaller region. Indeed, the algorithm ensures good geometry of the sample sets at each iteration. With other management of the sample sets, a bad ratio may occur because of bad geometry and to prevent this the model improvement algorithm is needed. Reducing the trust region radius with no information regarding the properties of the models (fully-linear or fully-quadratic) is likely to slow down the algorithm. It is especially important that the gradient of the model is related to the magnitude of Δ^k when the model gradient becomes small, otherwise the Taylor-like bounds do not guarantee the accuracy of the model and the convergence criterion on the model does not imply convergence on the true function.

2.3 The progressive barrier

We now provide the main components used by the PB to handle constraints. Let $\mathcal{V}^k$ denote the set of all points visited by the start of iteration k, and at which the values of the true functions, f and c, have previously been evaluated. The PB method uses a constraint violation function h: $\mathbb{R}^n \to \mathbb{R}^+ \cup \{+\infty\}$ which is an aggregation of the violations of the constraint functions c_i, $i \in \{1, 2, \ldots, m\}$. It satisfies $h(x) = 0$ if and only if x is feasible. For example, one can use $h_2 = \sum\limits_{i=1}^{m} (\max(c_i, 0))^2$, or $h_1 = \sum\limits_{i=1}^{m} (\max(c_i, 0))$, or even $h_\infty = \max\limits_{i \in \{1 \ldots m\}} (c_i, 0)$.

The PB relies on the barrier threshold h_{max}^k, which depends on k, the index of the iteration. The barrier threshold filters each point x for which $h(x) > h_{max}^k$. The idea is to obtain a feasible point by progressively decreasing the barrier threshold and selecting a point x at iteration k with a smaller constraint violation than at iteration $k-1$. However, the barrier threshold is not decreased too rapidly. Indeed, decreasing the threshold slowly allows selecting a promising infeasible point x with a low objective function value f. The sequence of thresholds $\{h_{max}^k\}_k$ is non-increasing.

The PB maintains two incumbent solutions at each iteration: a feasible x_F^k and an infeasible x_I^k one. The principle is to select these incumbents from $\mathcal{V}^k$ and to test nearby trial points in the hopes of improving the objective function value, f, or the constraint violation function value, h. These two points are defined at iteration k:

- The best feasible point, called the *feasible incumbent*:

$$x_F^k \in \operatorname{argmin}\{f(x) \ : \ x \in \mathcal{V}^k, \ h(x) = 0\}. \tag{4}$$

- The infeasible point within the barrier threshold with the least objective function value, called the *infeasible incumbent*:

$$x_I^k \in \operatorname{argmin}\{f(x) \ : \ x \in \mathcal{V}^k, \ 0 < h(x) \leq h_{max}^k\}. \tag{5}$$

An advantage of maintaining two incumbent solutions is that the local explorations around them could lead to different locally optimal solutions. This provides some sort of diversification algorithmic strategy. The exploration around the feasible incumbent often leads to a nearby minimizer. Furthermore, the exploration around the infeasible one allows displacements in the infeasible region.

At least one of these two incumbents exists. When both exist, the algorithm labels one of them as the *primary incumbent* x_P^k and the other as the *secondary incumbent* x_S^k:

$$\begin{array}{ll} x_P^k \leftarrow x_F^k \text{ and } x_S^k \leftarrow x_I^k & \text{if } f(x_I^k) \geq f(x_F^k) - \rho|f(x_F^k)|, \\ x_P^k \leftarrow x_I^k \text{ and } x_S^k \leftarrow x_F^k & \text{otherwise,} \end{array}$$

where $\rho > 0$ is a trigger, set at 0.1 in our computational experiments, to favor feasible over infeasible solutions. If only x_F^k or x_I^k exists, then x_P^k is defined with this one point and x_S^k is left undefined. The algorithm then explores around both incumbents, but deploys more effort around the primary one. In other words, more blackbox evaluations are computed near the incumbent that has the more promising objective function value.

If the infeasible incumbent has a lower objective function value than the feasible incumbent, i.e. $f(x_I^k) < f(x_F^k)$, then exploring near the infeasible incumbent may potentially lead to a feasible solution with a lower objective function value than $f(x_F^k)$.

Three types of iterations are distinguished, associated with different parameter update rules. These rules are based on the analysis of the set of points $\mathcal{V}^k$ and those evaluated during the iteration k. This set of points is $\mathcal{V}^{k+1}$.

- An iteration is said to be *dominating* whenever a trial point $x \in \mathcal{V}^{k+1}$ that dominates one of the incumbents is generated, i.e., when:

$$\begin{array}{rl} h(x) = 0 \text{ and } f(x) < f_F^k, & \text{or} \\ 0 < h(x) < h_I^k \text{ and } f(x) \leq f_I^k, & \text{or} \\ 0 < h(x) \leq h_I^k \text{ and } f(x) < f_I^k, & \end{array}$$

  where $f_F^k = f(x_F^k)$, $f_I^k = f(x_I^k)$ and $h_I^k = h(x_I^k)$. In this case, h_{max}^{k+1} is set to h_I^k.
- An iteration is said to be *improving* if it is not dominating, but if there is an infeasible solution $x \in \mathcal{V}^k$ with a strictly smaller value of h, i.e. when:

$$0 < h(x) < h_I^k \text{ and } f(x) > f_I^k.$$

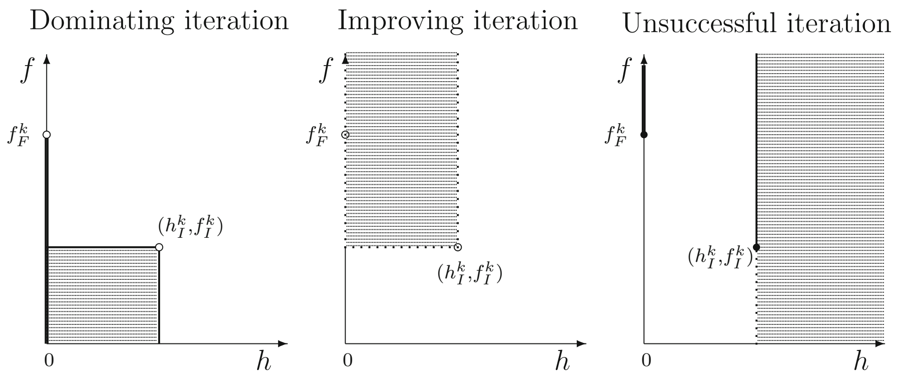

Fig. 1 Illustration of possible regions for the 3 different types of iterations. Figure adapted from [8]

In other words, there is an infeasible solution with a lower value of h than h_I^k but a higher value of f than f_I^k. The barrier threshold is updated by setting $h_{max}^{k+1} = \max\{h(x) \ : \ h(x) < h_I^k, \ x \in \mathcal{V}^k\}$. As a result, x_I^k is eliminated by the barrier threshold at iteration $k+1$. The points in $\mathcal{V}^{k+1}$ may have been generated during iteration k or during a previous iteration.

- An iteration is said to be *unsuccessful* when every trial point $x \in \mathcal{V}^k$ is such that:

$$h(x) = 0 \text{ and } f(x) \geq f_F^k, \quad \text{or} \quad h(x) = h_I^k \text{ and } f(x) \geq f_I^k \quad \text{or} \quad h(x) > h_I^k.$$

In this case, $h_{max}^{k+1} = h_I^k$ as in the dominating iteration. This implies that both incumbent solutions remain unchanged: $x_F^{k+1} = x_F^k$ and $x_I^{k+1} = x_I^k$. If the barrier threshold would be pushed further, no infeasible incumbent would be admissible as an infeasible current incumbent. Unlike the improving iteration, there are no other infeasible points to choose.

The improving iterations are the only ones which allow a reduction of the barrier threshold h_{max}^{k+1} below h_I^k. Figure 1 summarizes the three different cases. The leftmost figure illustrates a dominating iteration: a feasible trial point dominating x_F^k or an infeasible one dominating x_I^k is generated. The corresponding regions are represented by the thick line segment on the ordinate axis, and by the rectangular shaded region, respectively. The central figure illustrates an improving iteration: there is an $x \in \mathcal{V}^k$ whose image lies in the shaded rectangle: x has a lower constraint violation function value than h_I^k at the expense of a higher objective function value than f_I^k. Finally, the rightmost figure depicts an unsuccessful iteration. Every feasible point of $\mathcal{V}^k$ is dominated by the feasible incumbent, and every infeasible point of $\mathcal{V}^k$ has a higher constraint violation function value than h_I^k.

To summarize, convergence toward the feasible region is handled by the barrier threshold h_{max}^k, and selecting the infeasible point with the smallest objective function value aims at generating a solution with a good value of f.

2.4 The speculative line-search

In addition to the PB, we borrow a rudimentary line-search devised in the context of the MADS algorithm. The speculative search was first described in [7] as the *dynamic search*, and renamed as speculative search in [8]. The main idea of the speculative search is to explore further in a direction that leads to a dominating iteration.

More formally, in the context of the PB, let x^k be one of the incumbent solutions x_I^k or x_F^k. Suppose that exploration around x^k leads to a dominating iteration by generating the point x. Define $s = x - x^k$, the direction of success. The speculative search simply explores further along this direction at the sequence of trial points $x + 2s, x + 4s, \dots, x + 2^j s$ and stops as soon as a trial point does not dominate the previous one.

3 A derivative-free trust-region algorithm using the progressive barrier

The main idea of the new algorithm to treat Problem (1) is to combine the unconstrained DFTR algorithm of Sect. 2.1 with constraint handling techniques from the PB of Sect. 2.3. Models of the true functions f and c are built, and two constrained subproblems based on these models are minimized at each iteration (the exact statement of these subproblems appear in Step 3 of Algorithm 2).

3.1 Primary and secondary subproblems

The steps in the progressive barrier derivative-free trust-region (PBTR) algorithm are similar to those of DFTR, but there are two constrained subproblems around two incumbents: the primary subproblem around the primary incumbent and the secondary subproblem around the secondary incumbent. The constraint violation threshold is managed by the PB, as in the original version of PB for MADS. The efforts in term of blackbox evaluations are different around each incumbent. Building the models around one incumbent, the primary, is made by allowing more blackbox function evaluations than around the secondary incumbent. Section 4 details different implementations tested, with different strategies for the allocation of evaluations between the primary and the secondary subproblems.

Thus in comparison with Algorithm 1, that is for unconstrained problems, Step 0 is added to determine the primary and secondary incumbent solutions and Step 7 is added to explore further along directions that lead to dominating solutions, potentially generating new incumbent solutions for the next iteration. A minimal decrease on the objective function value f is imposed to accept new points.

Some additional modifications are introduced in this hybrid algorithm. As there are two subproblems at each iteration (one around each incumbent), we define two different trust region radii at each iteration: the trust region radius Δ_F^k around x_F^k and Δ_I^k around x_I^k. The notation $\tilde{c}_i^k$ is introduced to denote the models of the contraint function $c_i, i \in \{1, 2, \dots, m\}$. Furthermore, x_P^k, x_S^k denote the primary and secondary incumbents and $\hat{x}^k$, $\bar{x}^k$ denote the solution of the trust-region subproblems centred at the infeasible and feasible incumbents, respectively.

Recall that the set $\mathcal{V}^k \subset \mathbb{R}^n$ at iteration k corresponds to the points already evaluated by the start of iteration k. As before, the sample set of evaluated points used to build models around a point x^k is denoted by $\mathcal{Y}^k(x^k)$, and is built as presented in Sect. 2.1.

Algorithm 2 describes the kth iteration with the same algorithm parameters introduced in Sect. 2.1. Recall that when the geometry of the sample set is improved it guarantees a κ-fully-linear or a κ-fully-quadratic model depending on the type of model built.

Algorithm 2 PBTR for constrained optimization: Iteration k. Both x_F^k and x_I^k are assumed to exist. Otherwise, a single subproblem is considered.

Step 0 - Primary and secondary incumbents

Update x_F^k and x_I^k according to (4) and (5).
Let x_P^k be the primary incumbent and x_S^k be the secondary incumbent with $\{x_P^k, x_S^k\} = \{x_F^k, x_I^k\}$.

Step 1 - Construction of the sample set

The sample set $\mathcal{Y}^k(x_P^k)$ is built with $\frac{(n+1)(n+2)}{2}$ points as presented in Sect. 2.1 with geometry improvement. The sample set $\mathcal{Y}^k(x_S^k)$ is built with at least 2 points.

Step 2 - Models construction

Build certifiably κ-fully linear models $\tilde{f}_P^k$ and $\tilde{c}_P^k$ from $\mathcal{Y}^k(x_P^k)$ that approximate f and c in $B(x_P^k; \Delta_P^k)$.
Build models $\tilde{f}_S^k$ and $\tilde{c}_S^k$ from $\mathcal{Y}^k(x_S^k)$ that approximate f and c in $B(x_S^k; \Delta_S^k)$.
We have $\{\tilde{f}_F^k, \tilde{f}_I^k\} = \{\tilde{f}_P^k, \tilde{f}_S^k\}$ and $\{\tilde{c}_F^k, \tilde{c}_I^k\} = \{\tilde{c}_P^k, \tilde{c}_S^k\}$.

Step 3 - Subproblems

Optimize the pair of constrained subproblems on the trust-regions:

$$(S_I) \quad \begin{aligned} \hat{x}^k \in \underset{x\in\mathbb{R}^n}{\text{argmin}} \ & \tilde{f}_I^k(x) \\ \text{subject to } & \tilde{c}_I^k(x) \le 0 \\ & x \in B(x_I^k; \Delta_I^k) \\ & l \le x \le u \end{aligned} \qquad (S_F) \quad \begin{aligned} \bar{x}^k \in \underset{x\in\mathbb{R}^n}{\text{argmin}} \ & \tilde{f}_F^k(x) \\ \text{subject to } & \tilde{c}_F^k(x) \le 0 \\ & x \in B(x_F^k; \Delta_F^k) \\ & l \le x \le u. \end{aligned}$$

Step 4 - Step calculation

Evaluate f and c at $\hat{x}^k$ and $\bar{x}^k$.
Compute the ratios ρ_f^k and ρ_h^k as described in Sect. 3.2.

Step 5 - Update the barrier threshold h_{max}^{k+1}

The barrier threshold is updated by using PB principles and the classification of iteration in success, improvement and failure. See Sect. 3.2.

Step 6 - Update the trust region radii

The trust region radii Δ_F^k and Δ_I^k are updated following rules adapted from DFTR.
See Sect. 3.2.

Step 7 - Speculative line-searches

If $\hat{x}^k$ leads to a dominating iteration, perform a speculative search from x_I^k in the direction $s = \hat{x}^k - x_I^k$.
If $\bar{x}^k$ leads to a dominating iteration, perform a speculative search from x_F^k in the direction $s = \bar{x}^k - x_F^k$.
See Sect. 2.4.

3.2 Progressive barrier and trust-region update rules

The update rules for h_{max}, Δ_F^k and Δ_I^k are presented below. During iteration k, the true functions f and c are evaluated at $\hat{x}^k$ generated by solving Subproblem (S_I) and/or at $\bar{x}^k$ by solving Subproblem (S_F). Notice that the variables are differentiated by a hat and a bar rather than by subscripts F and I because it is possible that solving the subproblem whose domain is centered around the infeasible incumbent x_I^k might lead to a feasible solution, and solving the subproblem around x_F^k might lead to an infeasible solution. The barrier threshold h_{max}^{k+1} is updated as described in Sect. 2.3 with the following modification. The filtered set

$$\mathcal{F}^k = \left\{x \in \mathcal{V}^k : h(x) \leq h_{max}^k \text{ and } x \text{ is a non-dominated point in } \mathcal{V}^k\right\}$$

is built and used by the PB as the replacement for $\mathcal{V}^k$. This is done in order to use a more aggressive strategy than in MADS to reduce the value of the threshold. Indeed, since the new PBTR algorithm generates more points than MADS (in order to construct the models) and because the reduction of the barrier threshold is based on the h value of the history points, having more points in the history would make the barrier decrease too slowly.

The trust region radius update rules of DFTR are adapted to take the constraints into account and to improve both f and h. For the objective function, we compute the same ratio ρ_f^k as in DFTR. The ratio $\hat{\rho}_f^k$ compares the relative variation of the true function f and the model $\tilde{f}^k$ at $\hat{x}^k$ and x^k, and similarly $\bar{\rho}_f^k$ compares the evaluation of f at $\bar{x}^k$ and x^k:

$$\hat{\rho}_f^k = \frac{f(x^k) - f(\hat{x}^k)}{\tilde{f}^k(x^k) - \tilde{f}^k(\hat{x}^k)}, \qquad \bar{\rho}_f^k = \frac{f(x^k) - f(\bar{x}^k)}{\tilde{f}^k(x^k) - \tilde{f}^k(\bar{x}^k)}.$$

The model $\tilde{h}^k$ of the constraint violation function h is defined using the same norm, but with the model of c. For example, if $h = \sum_{i=1}^{m} (\max(c_i, 0))^2$, then $\tilde{h}_2^k = \sum_{i=1}^{m} \left(\max(\tilde{c}_i^k, 0)\right)^2$. The second set of ratios compare the variation of the true function h over the variation of the model function $\tilde{h}^k$:

$$\hat{\rho}_h^k = \frac{h(x^k) - h(\hat{x}^k)}{\tilde{h}^k(x^k) - \tilde{h}^k(\hat{x}^k)}, \qquad \bar{\rho}_h^k = \frac{h(x^k) - h(\bar{x}^k)}{\tilde{h}^k(x^k) - \tilde{h}^k(\hat{x}^k)}.$$

The new rules are an adaptation of the DFTR update rules, with three main differences. First, the ratio for the constraint violation function h is taken into account. Second, there are at most two incumbents: a feasible and an infeasible one (when the algorithm begins, there is only one incumbent, but as soon as both feasible and infeasible points have been identified, every iteration uses these two incumbents). Third, the condition to increase the trust region radius in Step 4 of Algorithm 1 is that the trust region radius is small compared to the norm of the model gradient of the objective function. In the constrained case, we introduce two Boolean variables defined as follows:

$$\mathcal{P}_I^k \text{ is true iff } \|\nabla \tilde{h}_I^k(x_I^k)\| \geq \mu \Delta_I^k \qquad \text{and} \qquad \mathcal{P}_F^k \text{ is true iff } \|\nabla \tilde{f}_F^k(x_F^k)\| \geq \mu \Delta_F^k$$

These Boolean variables are used to devise the trust region radius update rules. In what follows, the negation of $\mathcal{P}_I^k$ is denoted by $\neg\mathcal{P}_I^k$ and the negation of $\mathcal{P}_F^k$ is denoted by $\neg\mathcal{P}_F^k$.

For each trust region radius Δ_I^k and Δ_F^k the update rule depends on the status of feasibility of the new generated points $\hat{x}^k$ and $\bar{x}^k$ respectively. The update rules are similar to that of DFTR, with some modifications: the ratios for both f and h are taken into account when the new generated point is infeasible.

- Update rule for Δ_I^k when $\hat{x}^k$ is infeasible:

$$\Delta_I^{k+1} = \begin{cases} \gamma_{inc}\Delta_I^k & \text{if } \eta_1 \leq \hat{\rho}_f^k \text{ and } \eta_1 \leq \hat{\rho}_h^k \text{ and } \mathcal{P}_I^k, \\ \gamma_{dec}\Delta_I^k & \text{if } \hat{\rho}_f^k < \eta_0 \text{ or } \hat{\rho}_h^k < \eta_0 \text{ or } \neg\mathcal{P}_I^k, \\ \Delta_I^k & \text{otherwise.} \end{cases} \tag{6}$$

- Update rule for Δ_I^k when $\hat{x}^k$ is feasible:

$$\Delta_I^{k+1} = \begin{cases} \gamma_{inc}\Delta_I^k & \text{if } \eta_1 \leq \hat{\rho}_f^k \text{ and } \quad \mathcal{P}_I^k, \\ \Delta_I^k & \text{if } \eta_0 \leq \hat{\rho}_f^k < \eta_1 \text{ and } \mathcal{P}_I^k, \\ \gamma_{dec}\Delta_I^k & \text{otherwise.} \end{cases} \tag{7}$$

- Update rule for Δ_F^k when $\bar{x}^k$ is infeasible:

$$\Delta_F^{k+1} = \begin{cases} \gamma_{inc}\Delta_F^k & \text{if } \eta_1 \leq \bar{\rho}_f^k \text{ and } \eta_1 \leq \bar{\rho}_h^k \text{ and } \mathcal{P}_F^k, \\ \gamma_{dec}\Delta_F^k & \text{if } \bar{\rho}_f^k < \eta_0 \text{ or } \bar{\rho}_h^k < \eta_0 \text{ or } \neg\mathcal{P}_F^k, \\ \Delta_F^k & \text{otherwise.} \end{cases} \tag{8}$$

- Update rule for Δ_F^k when $\bar{x}^k$ is feasible:

$$\Delta_F^{k+1} = \begin{cases} \gamma_{inc}\Delta_F^k & \text{if } \eta_1 \leq \bar{\rho}_f^k \text{ and } \mathcal{P}_F^k, \\ \Delta_F^k & \text{if } \eta_0 \leq \bar{\rho}_f^k < \eta_1 \text{ and } \mathcal{P}_F^k, \\ \gamma_{dec}\Delta_F^k & \text{otherwise.} \end{cases} \tag{9}$$

In every case, when both ratios exceed η_0, but at least one of them is less than η_1, then the trust region radius remains unchanged at iteration $k+1$. Indeed, it is analogous to the unconstrained algorithm, in which a ratio between η_0 and η_1 implies no modification for the trust region radius.

When the new generated point is feasible, the update rules are defined by considering only the ratio of f, and similar rules as in the unconstrained case are applied.

4 Implementation and computational results

This section describes our Python implementation of the PBTR algorithm, and shows computational experiments comparing it with two state-of-the-art software packages. We first describe the implementation specifics of PBTR.

4.1 An implementation of the PBTR algorithm

Different options for the management of the sample sets are proposed. Finally a variant of the update rule for the barrier threshold, h_{max}, is presented, giving two options for this update, the original as in MADS and a new one.

4.1.1 Sample set management and subproblems

The techniques used to build the sample sets for linear and quadratic models are defined in Sect. 2.1. The PBTR algorithm distinguishes two incumbents, the primary and the secondary. Three different options are implemented to manage the sample sets of these incumbents. Each builds the sample set of points around a point x by taking every point at a distance within twice the size of the trust region radius. If that set of points is too large, the more recently generated ones are selected. Furthermore, at each iteration the geometry improvement algorithm is called, if necessary, for both the sample sets built around the primary incumbent and the secondary incumbent. The subproblems are optimized with a limit of 100 iterations of Ipopt used with a tolerance for each constraint equal to 10^{-8}, which is compliant with a global tolerance for the problem defined by $h_2(x) < 10^{-14}$. The solution produced by Ipopt does not necessarily satisfy the subproblem constraints.

The first option is named QUAD_QUAD. Quadratic models are built around both the primary and secondary incumbents x_P^k and x_S^k. The sample set around the primary iterate contains exactly $\frac{(n+1)(n+2)}{2}$ points and then the models built are completely determined. To have this exact number of points, new points are sampled and evaluated by respecting the well-poisedness of the sample set. The models are built with interpolation and are completely determined by the points in the sample sets. The sample set around the secondary iterate contains at most $\frac{(n+1)(n+2)}{2}$ points depending if there are less than $\frac{(n+1)(n+2)}{2}$ points in a ball of radius $2\Delta_S^k$ around the secondary iterate. Then the models built are underdetermined or completely determined depending of the number of points. In the underdetermined case the minimum of the Frobenius norm is taken as explained in [21, chap. 5]. More precisely, among all quadratic interpolation functions that pass through the sample set, we select the one with the least Frobenius norm of the Hessian matrix of the quadratic function.

The second option is named QUAD_LIN. Quadratic models are built around the primary incumbents and linear models around the secondary incumbents. As above, the primary sample set contains exactly $\frac{(n+1)(n+2)}{2}$ points and the models built are completely determined. The secondary sample set contains at most $n+1$ points depending if there are less than $(n+1)$ points in a ball of radius $2\Delta_S^k$ around the secondary iterate.

The third option is named LIN_LIN. Linear models are built around both primary and secondary incumbents x_P^k and x_S^k. The sample set around the primary iterate contains exactly $n+1$ points and then the models built are completely determined. The sample set around the secondary iterate contains at most $n+1$ points.

These three options for the management of the sample set are compared in Sect. 4.3.

4.1.2 Revised barrier threshold update rules

Recall that an improving iteration happens when at the end of an iteration, there are no points dominating the current incumbents, but there is at least one infeasible point which is not dominated by the infeasible incumbent. It means that there is at least one point x such that $f(x) > f(x_I^k)$ and $0 < h(x) < h(x_I^k)$.

A revised version of the barrier threshold update rule is proposed in case of an improving iteration. In the original version of the progressive barrier in MADS, as presented in Sect. 2.3, after an improving iteration k, the barrier threshold is set to the value $h_{max}^{original} = \max\{h(x) : h(x) < h_I^k,\ x \in \mathcal{V}^k\}$. Selecting $h_{max}^{original}$ to update the barrier threshold is appropriate in the context of MADS but not in the context of a DFTR algorithm. The reason is that to construct models, DFTR spends more function evaluations at each iteration than MADS, and the barrier threshold parameter would converge very slowly to zero.

To circumvent this undesirable behaviour, we define x_*^k as the infeasible point with the best value of h at the end of iteration k: $x_*^k \in \operatorname{argmin}\{h(x) : h(x) > 0,\ x \in \mathcal{F}^k\}$. and propose the following rule to reduce the threshold parameter: Following an improving iteration, we set

$$h_{max}^{k+1} = 0.9 \times h_{max}^{original} + 0.1 \times h(x_*^k).$$

This update rule guaranties to find at iteration $k+1$ an infeasible incumbent satisfying the barrier threshold. And it offers a trade-off to push the barrier threshold adequately, between a too aggressive strategy decreasing the barrier threshold too rapidly and an unaggressive strategy susceptible to fail to find a feasible point.

In the computational experiments below, the two barrier threshold update rules are labelled by ORIGINAL and REVISED.

4.2 Computational testbed

We now describe our computational testbed. The computational results are generated using 40 small-scale DFO analytical problems from the CUTEst collection [26] which respect the form of Problem (1), with inequality constraints and bounds. The unscaled problems from CUTEst are not selected because our implementation does not incorporate dynamic scaling as done in COBYLA from NLOPT and in NOMAD version 3.7.2 (such scaling will be the subject of future work).

For each analytical problem the initial point provided by CUTEst is used. It lies inside the bounds but does not necessarily satisfy the other constraints. Each instance is considered with a budget of $100(n+1)$ blackbox evaluations, the same order of magnitude used in [11] and [12].

The name, number of variables (n), number of constraints (m) and information about these problems are given in Table 1.

Two derivative-free multidisciplinary design optimization (MDO) problems from mechanical engineering are also used in a second round of computational experiments.

The first problem is called `AIRCRAFT_RANGE` and is taken from [43]. Computational experiments on this problem are conducted in [4,9,38]. Three coupled disciplines, namely structure, aerodynamics and propulsion are used to represent a simplified aircraft model with 10 variables. The objective function is to maximize the aircraft range under bounds constraints and 10 relaxable constraints. The blackbox implements a fixed point method through the different disciplines in order to compute the different quantities.

The second problem is called `SIMPLIFIED_WING` [44] and aims at minimizing the drag of a wing by optimizing its geometry through 7 bound-constrained variables, subject to 3 relaxable constraints. This multidisciplinary design optimization problem involves structures and aerodynamics. Both `AIRCRAFT_RANGE` and `SIMPLIFIED_WING` are initialized with a point chosen in the center of the region defined by the bounds.

Data profiles [37] are used to illustrate the results on the analytical problems. These graphs allow to compare different algorithms on a set of instances given a tolerance parameter $\tau \in [0, 1]$, fixed to 10^{-3} in this section. More precisely, a curve is associated to each method in a $x - y$ plane, where y corresponds to the proportion of problems close within τ to a reference solution, after x groups of $n + 1$ evaluations have been done. This reference solution is the best solution achieved by the different methods that are plotted in the graph. Data profiles were originally introduced for unconstrained problems, and have been adapted here to the constrained case, by considering only feasible solutions. With this strategy, it may occur though that no algorithm solves a problem when no feasible solutions have been found. A tolerance of 10^{-14} for h is used to consider a point as being feasible.

Performance profiles from [37] are also plotted. For such graphs, a performance ratio $r_{p,s}$ is defined by

$$r_{p,s} = \frac{t_{p,s}}{\min \left\{t_{p,s} \ : \ s \in S\right\}}$$

for Algorithm s on Problem p where S is the set of algorithms tested. It is the ratio of the number of evaluations needed to solve the problem ($t_{p,s}$) over the number of evaluations needed to solve it with the best algorithm. The performance profile $P_s(\alpha)$ of Solver $s \in S$ corresponds to the probability for s to have a performance ratio within a factor $\alpha \in \mathbb{R}$. It is approximated by

$$P_s(\alpha) = \frac{1}{n_p} size \left\{p \in \mathcal{P} \ : \ r_{p,s} \leq \alpha\right\}$$

Table 1 Description of the 40 analytical problems

Name	n	m	Lower bounds	Upper bounds	Initial point
avgasb	8	10	8	8	Feasible
b2	3	3	0	0	Infeasible
chaconn1	3	3	0	0	Infeasible
himmelp5	2	3	2	2	Infeasible
hs10	2	1	0	0	Infeasible
hs11	2	1	0	0	Infeasible
hs12	2	1	0	0	Feasible
hs15	2	2	0	1	Infeasible
hs18	2	2	2	2	Infeasible
hs19	2	2	2	2	Infeasible
hs22	2	2	0	0	Infeasible
hs23	2	5	2	2	Infeasible
hs24	2	3	2	0	Feasible
hs29	3	1	0	0	Feasible
hs30	3	1	3	3	Feasible
hs31	3	1	3	3	Feasible
hs33	3	2	3	1	Feasible
hs34	3	2	3	3	Feasible
hs35	3	1	3	0	Feasible
hs36	3	1	3	3	Feasible
hs43	4	3	0	0	Feasible
hs57	2	1	2	0	Feasible
hs64	3	1	3	0	Infeasible
hs72	4	2	4	4	Infeasible
hs76	4	3	4	0	Feasible
hs84	5	6	5	5	Feasible
hs86	5	10	5	0	Feasible
hs95	6	4	6	6	Infeasible
hs96	6	4	6	6	Infeasible
hs97	6	4	6	6	Infeasible
hs98	6	4	6	6	Infeasible
hs100	7	4	0	0	Feasible
hs101	7	6	7	7	Infeasible
hs108	9	13	1	0	Infeasible
kiwcresc	3	2	0	0	Infeasible
lootsma	3	2	0	1	Feasible
polak6	5	4	0	0	Infeasible
simpllpb	2	3	0	0	Infeasible
snake	2	2	0	0	Infeasible
spiral	3	2	0	0	Feasible

where n_p is the number of problems and $\mathcal{P}$ the set of problems. The curve on the far left side of a performance profile begins with $\alpha = 1$ and indicates the ratio of problems solved by s with the least number of evaluations. The curve when $\alpha \to \infty$ indicates on how many problems s converges. Data profiles and performance profiles are complementary.

Convergence graphs (objective value versus number of evaluations) are plotted for the two MDO problems. The NOMAD (version 3.7.2) and COBYLA (NLOPT) software packages are used with their default settings. NOMAD with default settings uses quadratic models as presented in [19]. The software COBYLA implements a derivative-free trust-region algorithm as described in [39]. The models used are linear and a penalty function is used to handle the constraint, based on the infinity norm.

4.3 Computational comparisons of strategies for PBTR

The testbed used is described in Sect. 4.2 and the different options implemented to test variants of PBTR have been described in Sect. 4.1. In addition, other solvers exist to solve Problem (1). Here computational results are presented to determine the best strategy for PBTR and its validity in existing algorithms.

Section 4.1 described three options for the sample set management and two options for the barrier threshold management. Recall that the names of the options for the sample set management are QUAD_QUAD, QUAD_LIN and LIN_LIN. And the names of the options for the barrier threshold management are ORIGINAL and REVISED. Figure 2 compares the six possible strategies by plotting the proportion of problems solved versus the number of groups of $n + 1$ evaluations. Figure 3 compares specifically the two quadratic strategies, QUAD_QUAD_ORIGINAL and QUAD_QUAD_REVISED, by plotting the proportion of problems solved versus the number of groups of $n + 1$ evaluations. The conclusions that are drawn do not take into account the time necessary to construct the linear and quadratic models. The figures reveal that combining quadratic models in both primary and secondary subproblems is the most efficient, and that the revised update rule for the barrier threshold performs better. Hence the best strategy is QUAD_QUAD_REVISED. The second best strategy also builds both quadratic models, but uses the original threshold update rule. Inspection of the six curves suggests that quadratic models are worth constructing, and the revised rule is always preferable to the original one, designed for another algorithm. At least for the problems tested, the improvements made at each iteration with quadratic models are worth the cost of building these models.

4.4 Comparison of PBTR with NOMAD and COBYLA

In order to validate our algorithm the best strategy for PBTR (QUAD_QUAD with REVISED), is compared to state-of-the art software packages NOMAD and COBYLA.

The data profiles in Fig. 4a shows that our algorithm is competitive with COBYLA on the benchmark set of smooth analytical problems. As expected, both model-based COBYLA and PBTRQUAD_QUAD perform better than the direct-search NOMAD algorithm. This behaviour was anticipated, as we do not recommend to use a direct-search

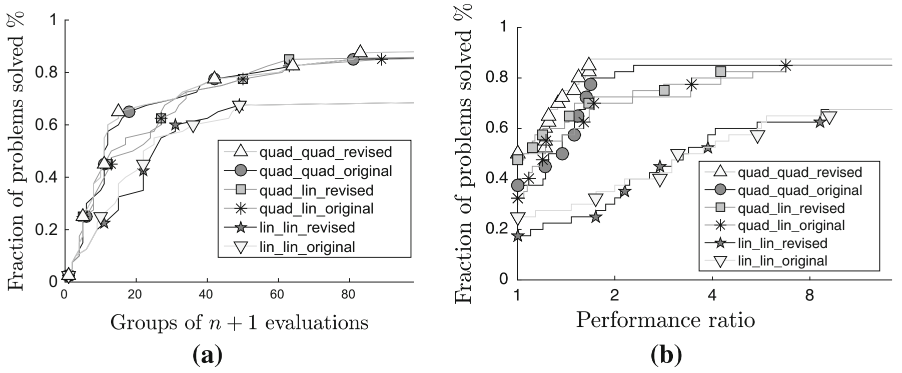

Fig. 2 Data profiles for six PBTR strategies. **a** Data profiles with $\tau = 10^{-3}$. **b** Perf. profiles at $100(n + 1)$ evaluations

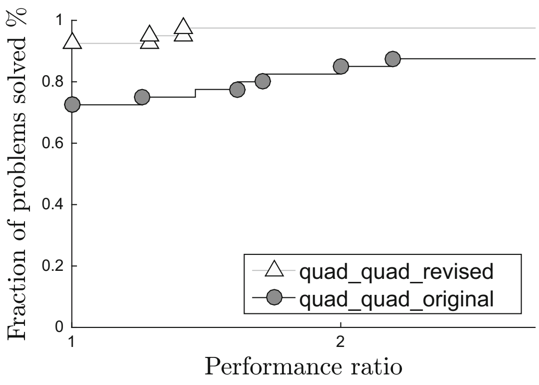

Fig. 3 Data profiles comparing the original and revised quadratic PBTR strategies

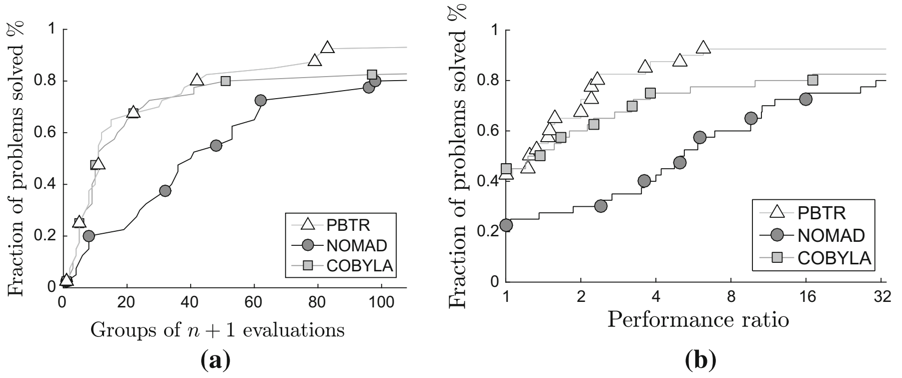

Fig. 4 Comparison of PBTR with NOMAD and COBYLA on DFO problems. **a** Data profiles with $\tau = 10^{-3}$. **b** Perf. profiles at $100(n + 1)$ evaluations

method for problems that can be well-approximated by smooth functions. The performance of PBTR is comparable to that of COBYLA, and when the number of function evaluations exceeds $40(n + 1)$, PBTR slightly outperforms COBYLA. This last observation is confirmed by the performance profiles in Fig. 4b.

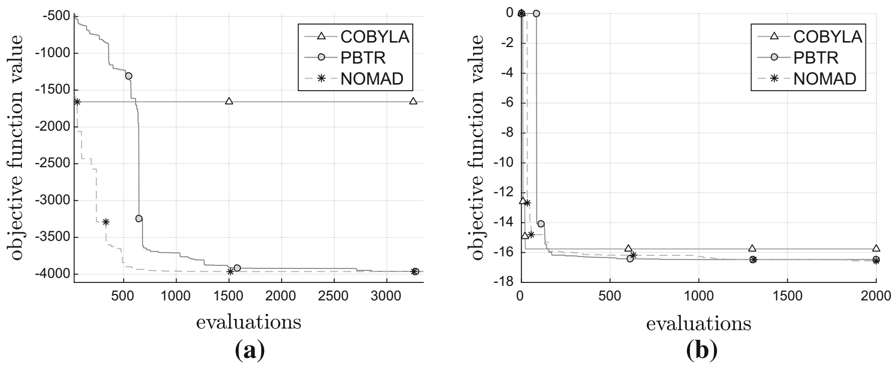

Fig. 5 Convergence graphs of PBTR, NOMAD and COBYLA for the two MDO problems. **a** AIRCRAFT_RANGE, **b** SIMPLIFIED_WING

The convergence graphs for both blackbox multidisciplinary design optimization problems are plotted in Fig. 5. For the AIRCRAFT_RANGE problem, the solver COBYLA decreases quickly but stalls at a feasible solution with objective function value around -1600, whereas both NOMAD and PBTR converge to solutions with a similar objective function value near -4000. This can be explained by the fact that COBYLA uses linear models. The plot reveals that NOMAD improves the solution more rapidly than PBTR on this BBO problem. The right part of the figure on the SIMPLIFIED_WING problem indicates a similar behaviour for COBYLA.

However, here both NOMAD and PBTR behave in a very similar way and both convergence graphs reach the same objective function value.

5 Discussion

This work shows how to treat nonlinear inequalities for derivative-free and blackbox optimization problems, by combining techniques from derivative-free trust-region methods with the progressive barrier strategy. After MADS, it is the first algorithm to deploy the PB.

Different strategies are compared and the best one is identified by computational results on a collection of 40 problems from the CUTEst collection. It consists of building quadratic models for every subproblem solved and of a trade-off rule to update the barrier threshold. This new algorithm PBTR combines features of both model-based and direct-search algorithms. Our computational results suggest that PBTR is competitive with COBYLA and preferable to NOMAD on analytical DFO problems, and that PBTR is competitive with NOMAD and preferable to COBYLA on nonsmooth blackbox optimization problems.

Future work includes the theoretical analysis of the proposed hybrid method, the integration of dynamic scaling in our implementation and other sample set managements, to allow overdetermined models, or to numerically analyze the frequency to improve the geometry of sample sets. Penalty function could also be examined for the subproblem treatment. Finally, prior to our work, the PB was only adapted to the MADS algorithm. We have shown that it can also be successfully adapted to a trust-region algorithm. Adaptations to other nonlinear algorithms, including derivative-based, should also be investigated.

Acknowledgements The authors would like to thank two anonymous referees for their careful reading and helpful comments and suggestions.

References

1. Abramson, M.A., Audet, C., Dennis Jr., J.E.: Filter pattern search algorithms for mixed variable constrained optimization problems. Pac. J. Optim. **3**(3), 477–500 (2007)
2. Arouxét, M.B., Echebest, N.E., Pilotta, E.A.: Inexact restoration method for nonlinear optimization without derivatives. J. Comput. Appl. Math. **290**, 26–43 (2015)
3. Audet, C.: A survey on direct search methods for blackbox optimization and their applications, chapter 2. In: Pardalos, P.M., Rassias, T.M. (eds.) Mathematics Without Boundaries: Surveys in Interdisciplinary Research, pp. 31–56. Springer, Berlin (2014)
4. Audet, C., Béchard, V., Le Digabel, S.: Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. J. Glob. Optim. **41**(2), 299–318 (2008)
5. Audet, C., Dennis Jr., J.E.: Analysis of generalized pattern searches. SIAM J. Optim. **13**(3), 889–903 (2003)
6. Audet, C., Dennis Jr., J.E.: A pattern search filter method for nonlinear programming without derivatives. SIAM J. Optim. **14**(4), 980–1010 (2004)
7. Audet, C., Dennis Jr., J.E.: Mesh adaptive direct search algorithms for constrained optimization. SIAM J. Optim. **17**(1), 188–217 (2006)
8. Audet, C., Dennis Jr., J.E.: A progressive barrier for derivative-free nonlinear programming. SIAM J. Optim. **20**(1), 445–472 (2009)
9. Audet, C., Dennis Jr., J.E., Le Digabel, S.: Globalization strategies for mesh adaptive direct search. Comput. Optim. Appl. **46**(2), 193–215 (2010)
10. Audet, C., Hare, W.: Derivative-Free and Blackbox Optimization. Springer Series in Operations Research and Financial Engineering. Springer International Publishing, Berlin (2017)
11. Audet, C., Ianni, A., Le Digabel, S., Tribes, C.: Reducing the number of function evaluations in mesh adaptive direct search algorithms. SIAM J. Optim. **24**(2), 621–642 (2014)
12. Audet, C., Le Digabel, S., Peyrega, M.: Linear equalities in blackbox optimization. Comput. Optim. Appl. **61**(1), 1–23 (2015)
13. Augustin, F., Marzouk, Y.M.: NOWPAC: a provably convergent derivative-free nonlinear optimizer with path-augmented constraints. Technical report, arXiv (2014)
14. Bandeira, A.S., Scheinberg, K., Vicente, L.N.: Convergence of trust-region methods based on probabilistic models. SIAM J. Optim. **24**(3), 1238–1264 (2014)
15. Bertsekas, D.P.: Constrained Optimization and Lagrangian Multiplier Methods. Academic, New York (1982)
16. Conejo, P.D., Karas, E.W., Pedroso, L.G.: A trust-region derivative-free algorithm for constrained optimization. Optim. Methods Softw. **30**(6), 1126–1145 (2015)
17. Conn, A.R., Gould, N.I.M., Toint, PhL: A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. SIAM J. Numer. Anal. **28**(2), 545–572 (1991)
18. Conn, A.R., Gould, N.I.M., Toint, Ph.L.: Trust-Region Methods. SIAM, MPS-SIAM Series on Optimization (2000)
19. Conn, A.R., Le Digabel, S.: Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. Optim. Methods Softw. **28**(1), 139–158 (2013)
20. Conn, A.R., Scheinberg, K., Vicente, L.N.: Global convergence of general derivative-free trust-region algorithms to first and second order critical points. SIAM J. Optim. **20**(1), 387–415 (2009)
21. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-Free Optimization. MOS-SIAM Series on Optimization. SIAM, Philadelphia (2009)
22. Custódio, A.L., Scheinberg, K., Vicente, L.N.: Methodologies and software for derivative-free optimization, chapter 37. In: Terlaky, T., Anjos, M.F., Ahmed, S. (eds.) Advances and Trends in

Optimization with Engineering Applications. MOS-SIAM Book Series on Optimization. SIAM, Philadelphia (2017)
23. Dennis Jr., J.E., Price, C.J., Coope, I.D.: Direct search methods for nonlinearly constrained optimization using filters and frames. Optim. Eng. **5**(2), 123–144 (2004)
24. Echebest, N., Schuverdt, M.L., Vignau, R.P.: An inexact restoration derivative-free filter method for nonlinear programming. Comput. Appl. Math. **36**(1), 693–718 (2017)
25. Fletcher, R., Leyffer, S.: Nonlinear programming without a penalty function. Math. Program. Ser. A **91**, 239–269 (2002)
26. Gould, N.I.M., Orban, D., Toint, Ph.L.: CUTEst: a Constrained and Unconstrained Testing Environment with safe threads for mathematical optimization. Comput. Optim. Appl. **60**(3):545–557 (2015). https://ccpforge.cse.rl.ac.uk/gf/project/cutest/wiki. Accessed 2015
27. Gould, N.I.M., Toint, PhL: Nonlinear programming without a penalty function or a filter. Math. Program. **122**(1), 155–196 (2010)
28. Gumma, E.A.E., Hashim, M.H.A., Ali, M.M.: A derivative-free algorithm for linearly constrained optimization problems. Comput. Optim. Appl. **57**(3), 599–621 (2014)
29. Kolda, T.G., Lewis, R.M., Torczon, V.: A generating set direct search augmented Lagrangian algorithm for optimization with a combination of general and linear constraints. Technical Report SAND2006-5315, Sandia National Laboratories, USA (2006)
30. Kolda, T.G., Lewis, R.M., Torczon, V.: Stationarity results for generating set search for linearly constrained optimization. SIAM J. Optim. **17**(4), 943–968 (2006)
31. Le Digabel, S.: Algorithm 909: NOMAD: nonlinear optimization with the MADS algorithm. ACM Trans. Math. Softw. **37**(4), 44:1–44:15 (2011)
32. Le Digabel, S., Wild, S.M.: A Taxonomy of Constraints in Simulation-Based Optimization. Technical Report G-2015-57, Les cahiers du GERAD (2015)
33. Lewis, R.M., Shepherd, A., Torczon, V.: Implementing generating set search methods for linearly constrained minimization. SIAM J. Sci. Comput. **29**(6), 2507–2530 (2007)
34. Lewis, R.M., Torczon, V.: Active set identification for linearly constrained minimization without explicit derivatives. SIAM J. Optim. **20**(3), 1378–1405 (2009)
35. Liuzzi, G., Lucidi, S.: A derivative-free algorithm for inequality constrained nonlinear programming via smoothing of an ℓ_∞ penalty function. SIAM J. Optim. **20**(1), 1–29 (2009)
36. Liuzzi, G., Lucidi, S., Sciandrone, M.: Sequential penalty derivative-free methods for nonlinear constrained optimization. SIAM J. Optim. **20**(5), 2614–2635 (2010)
37. Moré, J.J., Wild, S.M.: Benchmarking derivative-free optimization algorithms. SIAM J. Optim. **20**(1), 172–191 (2009)
38. Perez, R., Liu, H.H.T., Behdinan, K.: Evaluation of multidisciplinary optimization approaches for aircraft conceptual design. In: AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY, September (2004)
39. Powell, M.J.D.: A direct search optimization method that models the objective and constraint functions by linear interpolation. In: Gomez, S., Hennart, J.-P. (eds.) Advances in Optimization and Numerical Analysis. Mathematics and Its Applications, vol. 275, pp. 51–67. Springer, Dordrecht (1994)
40. Powell, M.J.D.: On fast trust region methods for quadratic models with linear constraints. Math. Program. Comput. **7**(3), 237–267 (2015)
41. Sampaio, PhR, Toint, PhL: A derivative-free trust-funnel method for equality-constrained nonlinear optimization. Comput. Optim. Appl. **61**(1), 25–49 (2015)
42. Sampaio, PhR, Toint, PhL: Numerical experience with a derivative-free trust-funnel method for nonlinear optimization problems with general nonlinear constraints. Optim. Methods Softw. **31**(3), 511–534 (2016)
43. Sobieszczanski-Sobieski, J., Agte, J.S., Sandusky Jr., R.R.: Bilevel integrated system synthesis. AIAA J. **38**(1), 164–172 (2000)
44. Tribes, C., Dubé, J.-F., Trépanier, J.-Y.: Decomposition of multidisciplinary optimization problems: formulations and application to a simplified wing design. Eng. Optim. **37**(8), 775–796 (2005)
45. Tröltzsch, A.: A sequential quadratic programming algorithm for equality-constrained optimization without derivatives. Optim. Lett. **10**(2), 383–399 (2016)
46. Xue, D., Sun, W.: On convergence analysis of a derivative-free trust region algorithm for constrained optimization with separable structure. Sci. China Math. **57**(6), 1287–1302 (2014)
47. Yuan, Y.: Recent advances in trust region algorithms. Math. Program. **151**(1), 249–281 (2015)
48. Yuan, Y.-X.: An example of non-convergence of trust region algorithms. In: Yuan, Y.-X. (ed.) Advances in Nonlinear Programming, pp. 205–215. Kluwer Academic, Dordercht (1998)