Globalization strategies for Mesh Adaptive Direct Search

Article *in* Computational Optimization and Applications · June 2010

DOI: 10.1007/s10589-009-9266-1

Comput Optim Appl
DOI 10.1007/s10589-009-9266-1

Globalization strategies for Mesh Adaptive Direct Search

Charles Audet · J.E. Dennis Jr. · Sébastien Le Digabel

Received: 5 November 2008

Abstract The class of Mesh Adaptive Direct Search (MADS) algorithms is designed for the optimization of constrained black-box problems. The purpose of this paper is to compare instantiations of MADS under different strategies to handle constraints. Intensive numerical tests are conducted from feasible and/or infeasible starting points on three real engineering applications.

The three instantiations are GPS, LTMADS and ORTHOMADS. Constraints are handled by the *extreme barrier*, the *progressive barrier*, or by a mixture of both. The applications are the optimization of a styrene production process, a MDO mechanical engineering problem, and a well positioning problem, and the codes are publicly available.

Work of the first author was supported by NSERC grant 239436-05. The second author was supported by LANL 94895-001-04 34, and the first two authors were supported by AFOSR FA9550-07-1-0302, the Boeing Company and ExxonMobil Upstream Research Company.

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

C. Audet
e-mail: Charles.Audet@gerad.ca
url: www.gerad.ca/Charles.Audet

S. Le Digabel
e-mail: Sebastien.Le.Digabel@gerad.ca
url: www.gerad.ca/Sebastien.Le.Digabel

J.E. Dennis Jr. (✉)
Computational and Applied Mathematics Department, Rice University, MS 134, 6100 South Main Street, Houston, TX 77005-1892, USA
e-mail: dennis@rice.edu
url: www.caam.rice.edu/~dennis

Published online: 17 June 2009

Keywords Mesh Adaptive Direct Search algorithms (MADS) · Black-box optimization · Constrained optimization · Nonlinear programming · Optimization test problems

1 Introduction

Black-box optimization is an useful term to describe a large class of realistic optimization problems. It occurs typically when the objective function and the functions defining the feasible region are evaluated by a computer code, a simulation, or an experiment. These evaluations are usually costly, even when they fail to return a value, and so the user wishes to use as few function calls as necessary. But there are other important aspects common to such problems that we will discuss later.

We write our optimization problems in the form

$$\min_{x \in \Omega} f(x), \tag{1}$$

where $\Omega = \{x \in X : c_j(x) \leq 0, j \in J\} \subset \mathbb{R}^n$, $f, c_j : X \to \mathbb{R} \cup \{\infty\}$ for all $j \in J = \{1, 2, \dots, m\}$, and where X is a subset of $\mathbb{R}^n$. Derivatives of f and of the c_j's are not explicitly available, and they are difficult to approximate with sufficient precision. The domain Ω is defined through the set X and the constraints $c_j(x) \leq 0$. This allows us to distinguish three types of constraints:

- *Unrelaxable constraints* define X and cannot be violated by any trial point. For example, the simulation can only be executed if the variables are positive.
- *Relaxable constraints* $c_j(x) \leq 0$ can be violated, and the simulation will execute, but $c_j(x)$ provides a measure of how much the constraint is violated.
- *Hidden constraints* [8] is a convenient term to exclude the set of points in the feasible region for the unrelaxable constraints at which the black-box fails to return a value for one of the problem functions. A typical example is when the simulation crashes unexpectedly.

Unrelaxable constraints must be satisfied by any design or decision for which the other constraints or the decision criteria are to be evaluated. They may not return a number at all, they may return only a message saying that the constraint is satisfied or not satisfied, i.e., x is in X or not. A user of our algorithms may also pose an unrelaxable constraint to restrict the search to a region where all interesting designs/decisions lie. For example, there may be bounds imposed on the decision variables that the designer wants to have always satisfied—not just at the solution. The feasible region for all such constraints defines X.

A hidden constraint is an unrelaxable constraint that *implicitly* excludes a portion of decision variable space that is feasible with respect to all the other constraints. Specifically, our algorithms must cope with the situation when the problem functions unexpectedly do not evaluate for no explicit reason. One tries to evaluate the design criteria or one of the other constraints, and the evaluation fails, usually at the same cost as when it succeeds. When this happens, we say that the variable $x \in X$ violates a hidden constraint, and we reject it. Thus, for both hidden and unrelaxable constraints, infeasible points are rejected. But, the unrelaxable constraints can be queried for feasibility before the relaxable constraints and objective function are called. This allows us to reject a point that fails to satisfy unrelaxable constraints at less computational expense than a point that fails a hidden constraint.

It might be tempting to think of hidden constraints as unrelaxable constraints that the user forgot to include in the problem definition. But that is far too simplistic. There are more fundamental reasons for the existence of unrelaxable constraints like the failure of a numerical method in the subroutine that evaluates one of the problem functions. For this reason, it has been traditional to provide information to the user on points that violate hidden constraints so they can try to tweak the evaluation subroutines to work for any such points they consider interesting.

This terminology concerning the constraints is also used in the recent book [9]. Similar ideas were also presented in Chap. 4 of [10].

A drastic way to handle constraints consists in rejecting any trial point that lies outside Ω. We call this the *extreme barrier* (EB) approach, and it was used by the Mesh Adaptive Direct Search (MADS) algorithm [3] to solve (1). MADS generalizes the pattern search (GPS) class of algorithms [20]. The first implementation of the MADS class of algorithms is called LTMADS [3]—LT since a random lower triangular matrix is involved in generating the exploration directions. It has two drawbacks that were treated separately in two subsequent papers.

First, for many instances of (1), finding a feasible point is a large part of solving the problem, and so a more subtle way to handle the relaxable constraints is given in [4]. A constraint violation function $h : X \to \mathbb{R}_+$ is used to measure the aggregate amount by which the relaxable constraints are violated. A threshold on h is imposed, and trial points whose constraint violation exceeds the threshold are rejected. As the algorithm unfolds, the threshold is reduced, but not necessarily at each step. We call this approach *the progressive barrier*, and the algorithm is MADSpb. It has the important advantage of allowing initial points, iterates, and trial points that violate the relaxable constraints.

A second drawback is that, while searching for improved solutions to (1), both implementations of MADS and MADS-PB were based on LTMADS, which uses randomness to generate trial directions. Consequently, runs could not be reproduced on different machines, and the quality of the solutions may vary significantly from one run to another. Furthermore, due to the way that the directions were generated, large regions of the space of variables could remain unexplored for several iterations. To remedy this, a quasi-random generator is used in [2] to generate orthogonal sets of directions, minimizing some measure of missed directions. The approach, called ORTHOMADS is deterministic and therefore can be reproduced on different machines. Moreover, the convergence result does not require a probabilistic argument as did the analysis of LTMADS.

This paper fills a gap in comparing three related ways to handle constraints with MADS. We consider GPS (with coordinate directions), LTMADS and ORTHOMADS, three instantiations of the MADS class of algorithms. In addition to EB and PB to handle the relaxable constraints we will present the *progressive-to-extreme* (PEB) approach, which consists in first treating the relaxable constraints with PB, and switching them to EB as the algorithm generates trial points that satisfy them. We will perform tests on three black-box test problems, from a feasible and an infeasible starting points, and also from both points simultaneously as the PB allows us to do. The three test problems consist of a styrene production simulation [5], a multidisciplinary design optimization from mechanical engineering [18, 19] and a groundwater supply and hydraulic capture community problem [14].

The paper is divided as follows: Sect. 2 briefly summarizes the GPS, LTMADS and ORTHOMADS algorithms, Sect. 3 discusses the EB, PB and PEB ways to handle the constraints, along with listing hierarchical convergence results based on the smoothness of the problem functions. Section 4 presents the three black-box test problems and specifies where an interested reader can obtain the software to run them. Section 5 illustrates numerically the new methods on the problems from three types of starting points.

2 Three instantiations of mesh adaptive direct searches

2.1 The MADS class of algorithms

MADS is an iterative algorithm in which the objective and constraints are evaluated at a finite number of trial points during each iteration. An iteration terminates when all selected trial points are evaluated, or it may be terminated as soon as a new incumbent solution is found. This is called the *opportunistic strategy*. At iteration k, the set of *feasible incumbent solutions* is defined to be

$$F_k = \left\{ \arg\min_{x \in V_k} \{ f(x) : x \in \Omega \} \right\},$$

where V_k denotes the set of trial points at which the black-boxes have been evaluated by the start of iteration k. V_0 is the user-defined set of starting points. The *feasible incumbent objective function value* is $f_k^F = f(x)$ for any $x \in F_k$. In order to ensure some convergence properties, all trial points must belong to a *mesh*. At iteration k, the mesh M_k is defined to be

$$M_k = \left\{ x + \Delta_k^m D z : x \in V_k, z \in \mathbb{N}^{n_D} \right\} \subset \mathbb{R}^n,$$

where $\Delta_k^m \in \mathbb{R}_+$ is the *mesh size parameter*, and D is a matrix in $\mathbb{R}^{n \times n_D}$ whose columns are n_D directions in $\mathbb{R}^n$. The parameter Δ_k^m is updated at the end of each iteration: It is decreased if and only if no trial point $t \in \Omega$ generated during the iteration satisfies $f(t) < f_k^F$. At every iteration, the mesh size parameter satisfies $\Delta_k^m = \Delta_0^m \tau^{r_k}$ where $\tau > 1$ is a fixed rational number, and r_k is an integer, which may be positive or negative. The matrix D is constant throughout all iterations, and must be the product of a non-singular $n \times n$ real matrix, with an integer $n \times n_D$ matrix.

Each iteration is composed of two steps: the *search* and the *poll*. The search step generates finitely many trial points on M_k anywhere in the space of variables. To do so, it can use the knowledge that the user has about the problem. It can use heuristics to attempt to move away from local solutions. It can use the surrogate management framework [6] and save a large amount of computational work by sampling on surrogate functions. A surrogate is typically a simplified version of the problem. It is less costly to evaluate, and can be used to guide the MADS algorithm [7].

The poll step follows more rigid rules, with trial points (*poll points*) satisfying some specific requirements: The set of directions D_k from the incumbent solution to the poll points must form a positive spanning set, each column of D_k must be a nonnegative integer combination of the directions in D, and furthermore, the distance from each poll point to the incumbent solution must be bounded above by a multiple of the *poll size parameter* Δ_k^p. The poll and mesh size parameters must satisfy $0 < \Delta_k^m \leq \Delta_k^p$, and must be such that if a subsequence of one of them goes to zero as k goes to infinity, then so does the same subsequence of the other. We achieve this by bounding above Δ_k^m to 1, and taking Δ_k^p to be $\sqrt{\Delta_k^m}$ when $\Delta_k^m < 1$. Otherwise ($\Delta_k^m = 1$), Δ_k^p takes values greater or equal than 1. More formally, the poll set is

$$P_k = \left\{x_k + \Delta_k^m d : d \in D_k\right\} \subset M_k,$$

where $x_k \in F_k$ is the *poll center* at iteration k.

At each poll step, a list of candidate trial points is generated. Some of the problems that we consider here have surrogates, and the surrogates are evaluated at that list of points. The list is then sorted by surrogate function value. The true function values are then evaluated first at the most promising poll points.

The next subsections briefly lists the main properties of three instantiations of MADS. The reader is invited to consult the appropriate papers for a complete description.

2.2 The GPS algorithm with coordinate directions

The simplest method considered in this paper is the GPS algorithm with coordinate directions [20]. Using the MADS notation given above, it can be described by setting $\Delta_k^p = \Delta_k^m$ and $D_k = [I_n \;\; -I_n]$ at every iteration k, where I_n is the $n \times n$ identity matrix.

2.3 The LTMADS algorithm

The LTMADS [3] implementation of the MADS algorithm generates the poll directions randomly. To achieve this, it generates a random integer non-singular lower triangular matrix (thus the LT in the name LTMADS), then it randomly permutes the rows. The columns of the resulting matrix B_k form a basis, and is completed to a maximal positive basis $D_k = [B_k \;\; -B_k]$. The set of normalized directions in the positive bases D_k becomes dense in the unit sphere with probability one as k goes to infinity.

2.4 The ORTHOMADS algorithm

The difference between ORTHOMADS [2] and LTMADS lies in the way that the poll directions D_k are generated. With LTMADS, they are generated randomly, and with ORTHOMADS, they are generated quasi-randomly, and are orthogonal to each other. The advantage of quasi-randomness is that runs can be reproduced in different computing environments. The advantage and disadvantage of randomness, as in LTMADS, is that it may lead to different solutions when launching multiple runs.

ORTHOMADS uses the quasi-random Halton [16] sequence $\{u_t\}^{\infty}$ in $\mathbb{R}^n$. This sequence is constructed by taking radical-base inverse functions of t in prime bases. At iteration k, there exists an integer ℓ such that the poll size parameter Δ_k^p is equal to $2^{-\ell}\Delta_0^p$. The integer ℓ is called the *mesh index* and may be positive or negative.

The absolute value of ℓ is used to obtain a Halton vector $u_t \in \mathbb{R}^n$. The vector $2u_t - e$ (where $e \in R^n$ is the vector of ones) is then scaled and rounded so that it is of appropriate length for the convergence analysis, and so that it is integer. The resulting vector is called $q_{t,\ell}$, the *adjusted Halton direction*, on which a Householder transformation is applied to generate an integer orthogonal basis of $\mathbb{R}^n$:

$$H_{t,\ell} = \|q_{t,\ell}\|^2 I_n - 2q_{t,\ell}q_{t,\ell}^T.$$

The maximal positive basis used by ORTHOMADS to construct the poll set is then $D_k = [H_{t,\ell} \; -H_{t,\ell}]$. The set of normalized directions in the positive bases D_k becomes dense in the unit sphere as k goes to infinity.

In [2], the Halton sequence is started at the Halton seed t_0, the n-th prime number: $\{u_t\}_{t=t_0}^{\infty}$. This removes linear correlation between some elements of the sequence. In the numerical tests of Sect. 5.4 we will perform various runs of ORTHOMADS with different values of the Halton seed t_0, and compare the results with multiple runs of LTMADS. This allows the generation of different reproducible ORTHOMADS runs.

3 Three strategies to handle constraints

In this section, we recall two strategies to handle constraints, and we propose a new one that combines both approaches.

3.1 The extreme barrier (EB)

In the original MADS paper [3], all constraints are handled by the extreme barrier approach. This can be interpreted as treating the problem as being unconstrained, by replacing the objective function $f(x)$ by

$$f_\Omega(x) = \begin{cases} f(x) & \text{if } x \in \Omega, \\ \infty & \text{otherwise.} \end{cases}$$

This strategy to handle constraint is not restricted in practice to our codes, indeed, it can be used on an *ad hoc* basis with virtually any algorithm. For example, both implementations KNITRO [21] and SNOPT [15] have this feature. This mechanism will often work well in practice, but is incompatible with the convergence analyses of the algorithms above, which rely on the assumption that the functions are smooth. Section 3.4 summarizes the MADS convergence analysis for nonsmooth functions.

Notice that this way of handling constraints forbids the use of starting points that do not satisfy the unrelaxable constraints. However, it is possible and easy to use the method in two phases, as described in Sect. 5.

3.2 The progressive barrier (PB)

MADS-PB [4] treats the relaxable constraints $c_j(x) \le 0$ differently than the unrelaxable ones defining X. It uses a nonnegative constraint violation function $h : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$:

$$h(x) := \begin{cases} \sum_{j \in J} (\max(c_j(x), 0))^2 & \text{if } x \in X, \\ \infty & \text{otherwise.} \end{cases}$$

Constraint violation functions in this context were first given by [11–13] for filter methods. A property of the constraint violation function is that $x \in \Omega$ if and only if $h(x) = 0$. Moreover, if $0 < h(x) < \infty$ then x satisfies the unrelaxable constraints but not all the relaxable ones: $x \in X \setminus \Omega$.

A threshold $h_k^{\max} \ge 0$ is imposed on the value of the constraint violation. The threshold depends on the iteration number k, and it follows a simple non-increasing rule given below. The threshold is used to reject any trial point $t \in \mathbb{R}^n$ generated at iteration k such that $h(t) > h_k^{\max}$. This is called the *progressive barrier* approach.

At iteration k, the set of feasible incumbent solutions, already defined in Sect. 2.1, is rewritten

$$F_k = \left\{ \arg\min_{x \in V_k} \{ f(x) : h(x) = 0 \} \right\},$$

and the set of *infeasible incumbent solutions* is defined to be

$$I_k = \left\{ \arg\min_{x \in U_k} \{ f(x) : 0 < h(x) \le h_k^{\max} \} \right\},$$

where U_k is the set of *infeasible undominated points*:

$$U_k = \{ x \in V_k \setminus \Omega : \nexists y \in V_k \setminus \Omega \text{ such that } y \prec x \}.^{1}$$

The feasible incumbent objective function value (f_k^F) equals $f(x)$ for any $x \in F_k$. If $F_k = \emptyset$, then f_k^F is set to ∞. The *infeasible incumbent objective* and *constraint violation function values* are denoted by f_k^I and h_k^I, and equal $f(x)$ and $h(x)$ for any x in I_k. If $I_k = \emptyset$, then h_k^I is set to ∞, and f_k^I to $-\infty$. The reason for setting f_k^I to that value is explained at the end of this subsection.

At iteration k, polling is done around elements of both F_k and I_k, called *primary and secondary poll centers*. The primary poll center is taken in I_k if $f_k^I < f_k^F - \rho$, where ρ is a user defined parameter (our tests use the default value $\rho = 0.1$). Otherwise the primary poll center is taken in F_k. Polling is done more extensively around the primary poll center than around the secondary poll center. Complete details can be found in [4].

If a feasible trial point $t \in \Omega$ satisfying $f(t) < f_k^F$ is generated, or if an infeasible trial point $t \in X$ satisfying $t \prec x$ for any $x \in I_k$ is generated, then the iteration is called *dominating*. If an iteration is not dominating, but an infeasible trial point $t \in X$

[1] $y \prec x$ signifies that $h(y) < h(x)$ and $f(y) \le f(x)$, or that $h(y) \le h(x)$ and $f(y) < f(x)$.

satisfying $h(t) < h_k^I$ is generated, then the iteration is called *improving*. Otherwise the iteration is labeled as being *unsuccessful*. The opportunistic strategy mentioned in Sect. 2.1 can only interrupt dominating iterations.

We now explain why f_k^I is set to $-\infty$ when $I_k = \emptyset$: Consider the case where $I_k = \emptyset$ and an infeasible trial point is generated during the iteration. If f_k^I was equal to $-\infty$, then the iteration would be dominating, and would be allowed to stop immediately. Setting $f_k^I = -\infty$ makes the iteration improving and forces the algorithm to explore other trial points before terminating the iteration. In other words, the algorithm will not settle for the first infeasible trial point that it generates.

The barrier threshold parameter is updated at the end of iteration k as follows:

$$h_{k+1}^{\max} = \begin{cases} \max_{y \in V_{k+1}} \{h(y) : h(y) < h_k^I\}, & \text{if iteration } k \text{ is improving,} \\ h_k^I, & \text{otherwise.} \end{cases} \tag{2}$$

An advantage of PB over EB is that the former allows the use of infeasible starting points.

3.3 The progressive-to-extreme barrier (PEB)

In case none of the initial points are feasible, but one wishes to use the extreme barrier approach in a single phase, we introduce a third strategy to handle the relaxable constraints: The *progressive-to-extreme barrier* (PEB) strategy consists in initially treating a relaxable constraint by the progressive barrier. If polling around the infeasible poll center generates a new infeasible incumbent that satisfies a constraint violated by the poll center, then that constraint moves from being treated by the progressive barrier to the extreme barrier. This will not affect the value of h at that trial point, but notice that some undominated trial points in U_k might now have a value of h equal to infinity. These points will no longer be in U_k. In fact, the set U_k must then be reconstructed from the set of previously evaluated points. With this approach, it is possible for all the relaxable constraints to be treated after finitely many iterations by the extreme barrier.

3.4 Convergence analysis

Theoretical convergence of LTMads-EB, OrthoMads-EB and LTMads-PB are analyzed in detail in [2, 3] and [4], respectively. The convergence analysis of PB is independent of the instantiation of Mads, and therefore can be applied directly to OrthoMads. Furthermore, the convergence analysis of OrthoMads-PB is identical to that of LTMads-PB, except that the results hold without any probabilistic arguments. We omit the proofs of the results stated in this subsections since they are identical to the analogous results shown in [2, 4].

The convergence of PEB is identical to that of EB when all relaxable constraints change status and end up being treated by the EB. Otherwise the analysis is identical to that of PB. The convergence analysis relies on the following assumptions, taken from [4]:

A1: There exists some point x_0 in the user-provided initial set V_0 such that $x_0 \in X$, and $f(x_0)$ and $h(x_0)$ are both finite.

A2: All trial points considered by the algorithm lie in a bounded set.
A3: For every hypertangent direction $v \in T^H_\Omega(\hat{x}) \neq \emptyset$, there exists an $\epsilon > 0$ for which the Clarke generalized derivative of h satisfies

$$h^\circ(x; v) < 0 \quad \text{for all } x \in \{x \in X \cap B_\epsilon(\hat{x}) : h(x) > 0\}.$$

Under the PEB approach, the function h used in the following convergence analysis is not necessarily the one that was used at iteration 0. While the algorithm was applied, some constraints may have changed status from progressive to extreme. The h used in the analysis is the one that was used at infinitely many iterations.

Under assumptions A1 and A2, applying ORTHOMADS-PB or ORTHOMADS-PEB to problem (1) generates a convergent refining subsequence $\{x_k\}_{k\in K}$. Each member of such a subsequence is such that iteration $k \in K$ is unsuccessful, and the subsequence of mesh size parameters $\{\Delta^m_k\}_{k\in K}$ converges to 0. The poll center x_k is said to be a *minimal frame center*. The limit of a convergent refining subsequence is called a *refined point* $\hat{x}$. In summary, the algorithm generates a refined point $\hat{x}$, i.e., the limit of a subsequence of minimal frame centers on meshes that get infinitely fine. This result holds without any assumption on the smoothness of the functions defining the problem.

The convergence analysis may be pushed further by assuming more on the differentiability of f and on the nature of the tangent cones. We refer the reader to [3] for definitions and pointers to original definitions of *Clarke derivatives*, *strict differentiability*, *regularity* and *contingent cones*. With these notions, the hierarchy of convergence results extends to:

(i) If $\hat{x} \in \Omega \subseteq X$ and if f is Lipschitz near $\hat{x}$, where $T^H_\Omega(\hat{x}) \neq \emptyset$, and if the refining subsequence contained infinitely many feasible poll centers,[2] then $\hat{x}$ is a Clarke stationary point for the minimization of f over Ω.
(ii) In addition to (i), if f is strictly differentiable at $\hat{x}$, then $\hat{x}$ is a Clarke KKT stationary point for the minimization of f over Ω.
(iii) In addition to (i), if Ω is regular at $\hat{x}$, then $\hat{x}$ is a contingent stationary point for the minimization of f over Ω.
(iv) If (ii) and (iii) hold, then $\hat{x}$ is a contingent KKT stationary point for the minimization of f over Ω.

Analogous results hold for the minimization of the constraint violation function over the domain Xdefined by the unrelaxable constraints. These results apply in particular when the PEB approach did not change the status of the relaxable constraints from PB to EB.

(v) If $\hat{x} \in X$ and if h is Lipschitz near $\hat{x}$, where $T^H_X(\hat{x}) \neq \emptyset$, then $\hat{x}$ is a Clarke stationary point for the minimization of h over X.
(vi) In addition to (v), if h is strictly differentiable at $\hat{x}$, then $\hat{x}$ is a Clarke KKT stationary point for the minimization of h over X.

[2]Assumption A3 is sufficient, but not necessary, to ensure the existence of infinitely many feasible poll centers.

(vii) In addition to (v), if X is regular at $\hat{x}$, then $\hat{x}$ is a contingent stationary point for the minimization of h over X.

(viii) If (vi) and (vii) hold, then $\hat{x}$ is a contingent KKT stationary point for the minimization of h over X.

The above hierarchical convergence results hold for both OrthoMads-PB and OrthoMads-PEB. They also hold for LTMads-PB and LTMads-PEB, but are true with probability one since these instantiations of Mads use randomness to generate the poll directions (see [4] for the detailed analysis of LTMads-PB).

4 Test problems

This section tests the various algorithms on three problems which correspond to real applications based on non-trivial computer simulations. These black-box problems are the sort for which direct search methods like Mads are designed. For example, there are often hidden constraints in such problems. Pointers to the problem source codes are available on the Nomad website [1].

4.1 Styrene production simulation (Styrene problem)

This simulation, proposed in [5], uses some common methods such as Runge-Kutta, Newton, Wegstein (fixed points), secant, bisection, and many other chemical engineering related solvers. During our numerical experiments, we have observed that for some values of x, the simulation returns different values on different computing environments, and about 14% of the time the trial points violate a hidden constraint. Numerical results for the Styrene problem presented in this work are reproducible under the Mac OS 10.5.5 environment.

The objective of this problem is to maximize net present value (NPV) while satisfying industrial and environmental regulations. There are 8 bound constrained variables, 4 boolean unrelaxable constraints, and 7 relaxable constraints. We minimize the negative of the NPV.

A surrogate of the simulation is obtained by using greater tolerance values and smaller maximum numbers of iterations in the various numerical methods. Each evaluation of the true simulation requires approximatively the same CPU time of three surrogate evaluations. In our results, the number of evaluations is the sum of number of true evaluations, with a third of the number of surrogate evaluations. For comparison purposes, a total of 1000 evaluations was used. Starting points are shown in Table 1.

Table 1 Starting points for the Styrene problem

Feasible	$(54, 66, 86, 8, 29, 51, 32, 15)$	$f(x) = -10942600$
Infeasible	$(44, 99, 76, 39, 39, 48, 43, 5)$	$f(x) = -1932060$

Table 2 Starting points for the MDO problem

Feasible	(0.4, 1, 0.872, 0.4433, 0.05, 45000, 1.728, 3.196, 62.68, 1000)	$f(x) = -703.57$
Infeasible	(0.4, 0.75, 0.75, 0.189296875, 0.09, 57000, 1.4, 2.5, 70, 1500)	$f(x) = -3758.86$

4.2 Multidisciplinary design optimization (MDO problem)

This problem is a multidisciplinary design optimization (MDO) problem taken from [18, 19] from the mechanical engineering literature. It is also studied in [5]. Three coupled disciplines (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 black-box implements a fixed point iteration through the different disciplines in order to compute the aircraft range. We minimize the negative of the range.

A natural surrogate consists in having the simulation use a greater relative error on the linking fixed point iterations and a smaller limit on the maximal number of fixed point iterations as stopping criteria. The total number of fixed point iterations used by the surrogate and true functions evaluations is used as a stopping criteria. For comparison purposes, we allowed a total of 10,000 fixed point iterations, which corresponds to about 650 function evaluations. Starting points are shown in Table 2.

We generated the infeasible starting point by minimizing the objective subject only to the unrelaxable bound constraints. This results in an infeasible starting point with a much lower objective value than the feasible starting point, as is generally the case for real engineering problems. Such a starting point is a good test of our constraint handling techniques since the optimizer must drag the incumbent solution uphill (with respect to the objective) to feasibility. The results show that we are not able to accomplish this in all the tests.

4.3 Well placement (Well problem)

Several well placement problems are described in [14]. The objective is to minimize the cost needed to prevent an initial contaminant plume from spreading by using wells to control the direction and extent of advective fluid flow. We will study the problem with six wells and nonlinear head constraints.

Some numerical experimentation revealed that the objective function was very sensitive to the sum of the pumping rate values. The objective function value deteriorates rapidly as the sum of the pumping rates differs from the minimum allowable total extraction rate. Therefore, we replaced the linear constraint (12) of [14] by an equality, and used it to eliminate the pumping rate of the sixth well as an explicit variable.

The 17 bound constrained decision variables are the well locations $\{(x_i, y_i)\}_{i=1}^{6}$, and pumping rates $\{Q_i\}_{i=1}^{5}$.

Evaluation of the objective function value requires running a Fortran solver to simulate the groundwater flow. In the original paper [14], all constraints were handled

Table 3 Starting points for the Well problem

Feasible	$(48, 21, 44, 9, 39, 35, 33, 23, 26, 23, 14, 12, -0.005333, -0.005333, -0.005333, -0.005333, -0.005333)$	$f(x) = 165403.81$
Infeasible	$(14, 27, 36, 17, 28, 16, 42, 3, 37, 18, 41, 19, -0.006176218166203, -0.005318565977194, -0.004752139653219, -0.006100217986489, -0.00593769420205)$	$f(x) = 168444.26$

directly by the black-box, except for the twelve bound constraints on the allowable head (the non-linear constraints (11) of [14]). All constraints were unrelaxable and treated by EB. In the tests conducted here, these twelve constraints are handled by the EB, PB or PEB approaches. Bound constraints on the sixth pumping rate are treated by the EB approach. This results in trial points that encounter hidden constraints about 6% of the time.

Starting points are shown in Table 3. The feasible starting point has a better objective value than the infeasible starting point, and this would seem to favor the EB approach. However, we were unable to find an infeasible starting point that caused any problem. We conclude that the difficulty in this problem for the methods tested here is not in dealing with infeasible starting points.

The termination criteria for the well placement problem is fixed to 1000 function evaluations.

5 Numerical results

We have presented three algorithms and three strategies to handle relaxable constraints. These nine possibilities are discussed in the next subsections. The first subsection summarizes the results from a feasible starting point only, the second from an infeasible starting point only, and the third when both feasible and infeasible starting points are available to start the two progressive barrier variants—we did not make these runs for the EB approach since they are redundant with the ones from the feasible starting point. The fourth subsection compares multiple runs of LTMADS and ORTHOMADS using different seeds. This might be a useful option in a massively parallel environment. In the fifth subsection, we give the best solutions found for all three problems in all our runs. We identify which algorithm and strategy produced the best result and whether or not it is reproducible.

All EB runs from an infeasible point are performed in two phases: First, minimization of the constraint violation h over X is conducted and terminated as soon as a feasible point is generated. Second, problem (1) is then solved from the feasible point found in the first phase. In some of our numerical experiments, the first phase failed to produce a feasible point.

All tests have been conducted with the version 3.0 of the NOMAD solver [1], with all parameters set to the default values. The poll step always uses $2n$ directions around the primary poll center. In GPS, the secondary directions are the $n + 1$ directions: the coordinate directions $\pm e_i$, where $i = 1, 2, \dots, n$, and the negative of the sum of these directions. For LTMADS and ORTHOMADS, 2 opposite directions are used around the secondary poll center; in LTMADS they are $\pm b(\ell)$ (see [3]), and in ORTHOMADS they are $\pm q_{t,\ell}$ (see [2]). The poll is interrupted opportunistically as soon as a new incumbent solution is generated. The reason we used more secondary poll directions for GPS is as follows. For ORTHOMADS, we can prove that the normalized secondary poll directions become asymptotically dense in the unit sphere. For LTMADS, the same result holds with probability one. Such a result is not possible using only coordinate directions, and so for GPS, we felt that we should use a minimal positive basis of secondary polling directions.

An important point here is that the search step was not used to increase efficiency. When these methods are used to solve expensive problems, a great deal of cost savings can occur from using searches well suited to the problem. Some convincing statistics on this point are given in [17]. We put no effort into finding effective searches because the main issues treated here concern different ways of polling and of handling constraints. These issues would be obscured by confounding them with search techniques. We are much more concerned with how polling and constraint handling behave in finding good solutions. Thus, we also run the algorithms past a point where a user might stop them. We then use the amount of work used by the different methods to compare their performance, but only in the context of finding good solutions from feasible and infeasible starting points.

Sections 5.1 to 5.3 compare single runs of GPS, LTMADS and ORTHOMADS. The outcome of the runs are illustrated in Figs. 1 to 9. The plots on the first row of each figure depict the progress of the incumbent feasible objective function value against effort expended. Note that for the MDO problem, the abscissa is the total number of fixed point iterations used by the surrogate and by the full accuracy simulation. This is much more than the number of full accuracy simulations, which each require multiple fixed point iterations.

The plots in the first column correspond to EB, the second column to PB, and the third column to PEB. The plots in the last row map f versus $h^{1/4}$ for all trial points of the best run under EB, PB, and PEB ($h^{1/4}$ is used instead of h for readability). A boxed label indicates which method produced the best final objective function value.

This first phase of tests intends to mimic a specific utilization of the algorithms, where only one run can be conducted, because of the cost of the functions to evaluate, or simply because users will only try a method once for a given problem.

Multiple runs are performed in Sect. 5.4 for the second phase of tests. The purpose is to compare the efficiency of LTMADS and ORTHOMADS without the influence of the random and Halton seeds. A total of 30 LTMADS runs, with different random seeds, are performed for each combination of constraint treatment and starting point types. The randomness of LTMADS prevents reproducibility of these experiments on different machines. For comparison, 30 reproducible runs of ORTHOMADS with different Halton seeds are also performed. This sums up to a total of 1440 runs in addition to 24 GPS runs in the first testing phase.

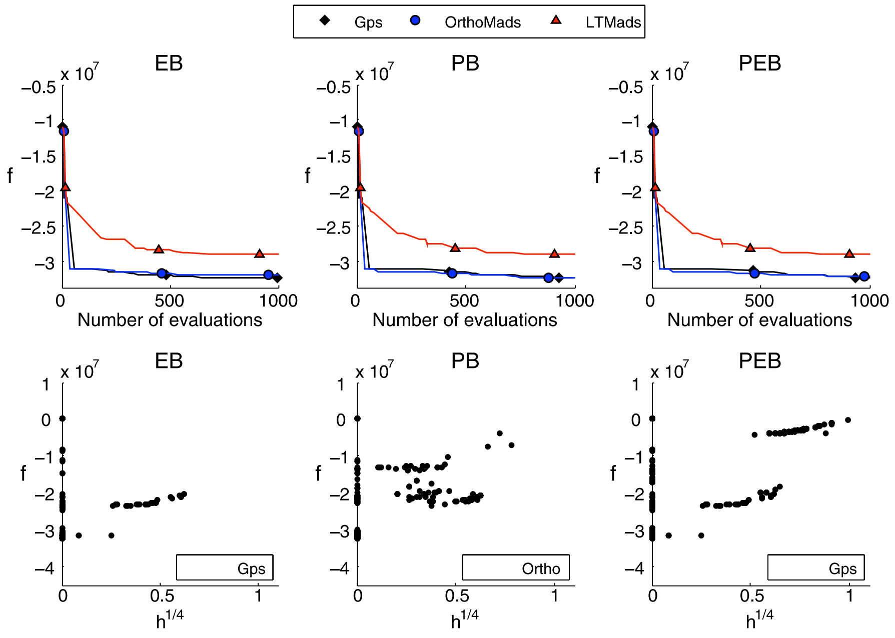

Fig. 1 Styrene problem from a feasible starting point

5.1 A feasible initial point

Figure 1 describes the runs for the Styrene problem from a feasible starting point. There is not an important difference in the final objective function value between the ways that constraints are handled. However, GPS and ORTHOMADS behave in a similar way and clearly outperform LTMADS. The plots of f versus $h^{1/4}$ show that PB is the constraint handling strategy that best traverses the infeasible region for the relaxable constraints.

In Fig. 2 below, ORTHOMADS is the most stable method. In all three runs, it reached the best known solution, and twice GPS and LTMADS failed to approach it. An interesting aspect of this figure is the GPS-PB run. A closer look at the log shows that the secondary poll strategy was key to the excellent result.

As for the Styrene problem, runs for the Well problem from a feasible starting point are very similar one to another when viewed from the f *versus number of evaluations* perspective. The GPS algorithm reached slightly better solutions in all three cases, which supports the implications in [14] that the coordinate directions are quite effective for these problems. The EB strategy works very well for this problem with all three algorithms.

5.2 An infeasible initial point

In the plots of Fig. 4 for the Styrene problem from an infeasible point, the strategy ORTHOMADS-PB outperforms all the others. The plot of f versus $h^{1/4}$ suggests that

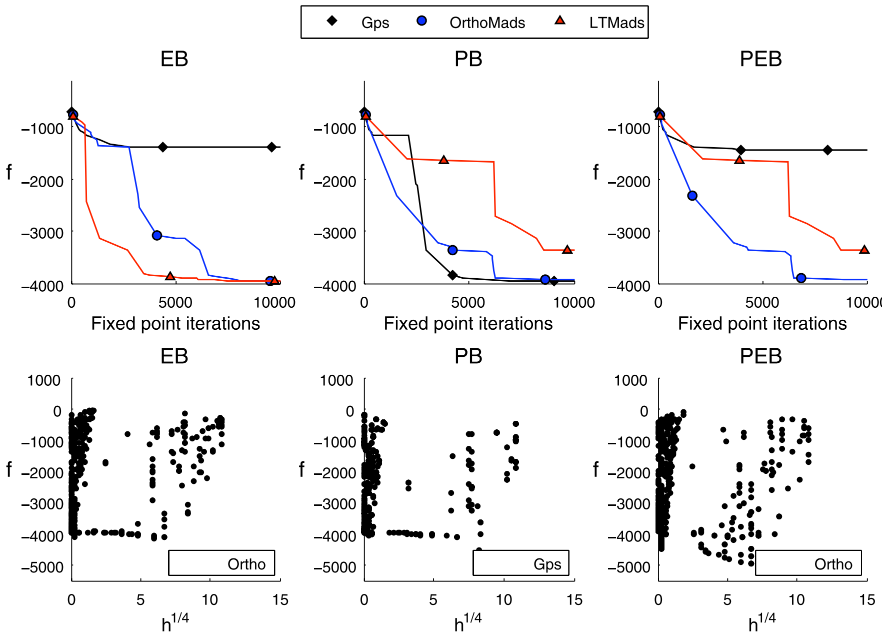

Fig. 2 MDO problem from a feasible starting point

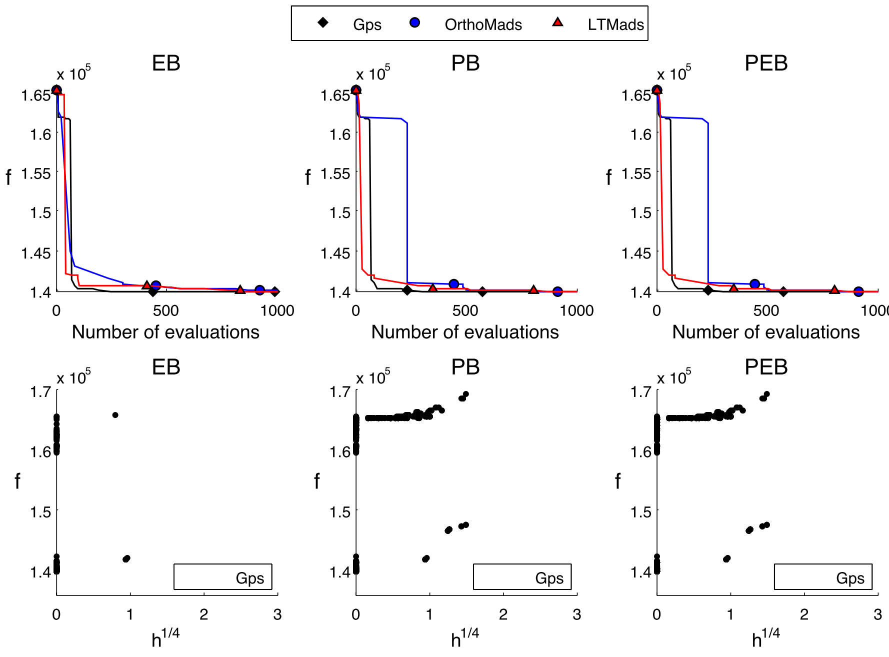

Fig. 3 Well problem from a feasible starting point

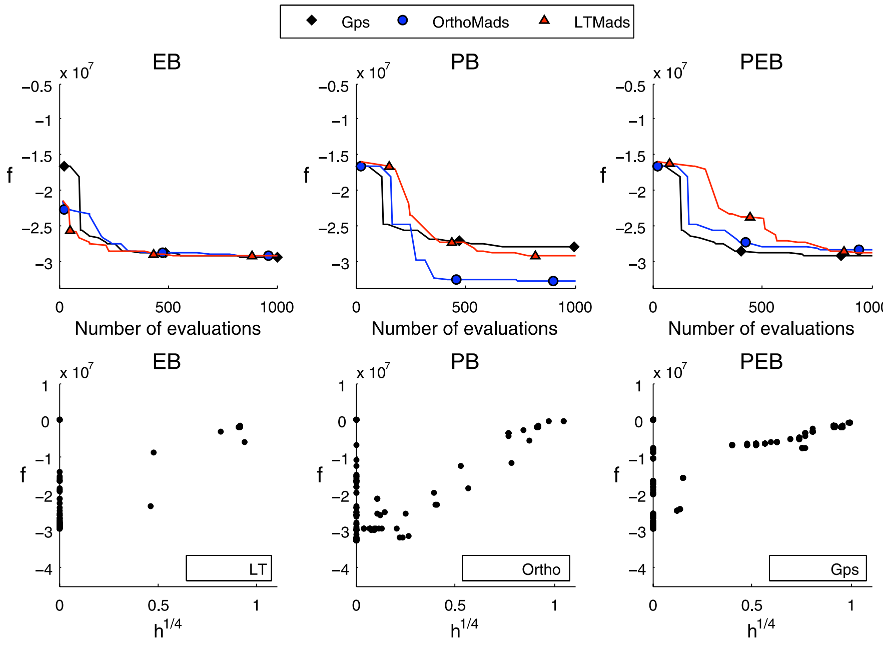

Fig. 4 Styrene problem from an infeasible starting point

ORTHOMADS-PB was able to generate a better solution by exploring the infeasible region differently.

A good combination for the MDO problem from an infeasible starting point illustrated in Fig. 5 appears to be PB, especially ORTHOMADS-PB. The blank graph for the EB is there because *no* algorithm coupled with EB was able to generate a feasible point, and so there is nothing to plot. There are no plots at all for GPS, because it did not generate any feasible point using either of EB, PB, or PEB. Note that the only good run for GPS on the MDO problem for either feasible or infeasible starting guesses was GPS-PB from a feasible starting point. A careful examination of the numerical output reveals that a couple of the $n + 1$ point secondary polling steps succeeded very well for the good GPS-PB run. We see this as justification for doing the additional work of secondary polling.

Figure 6 for the Well problem from an infeasible starting point shows that all methods with all strategies reached the best known value. EB, and especially GPS-EB, appears to be a positive strategy for this problem.

5.3 A feasible and an infeasible initial points

From both feasible and infeasible starting points, Fig. 7 for the Styrene problem suggests that PB and PEB are about the same. ORTHOMADS and GPS perform very well, with an advantage in the decrease rate to GPS. LTMADS did not achieve a good solution.

As expected, GPS does very well on the MDO problem using both starting points as illustrated in Fig. 8. Remember that GPS-PB with a feasible starting point was

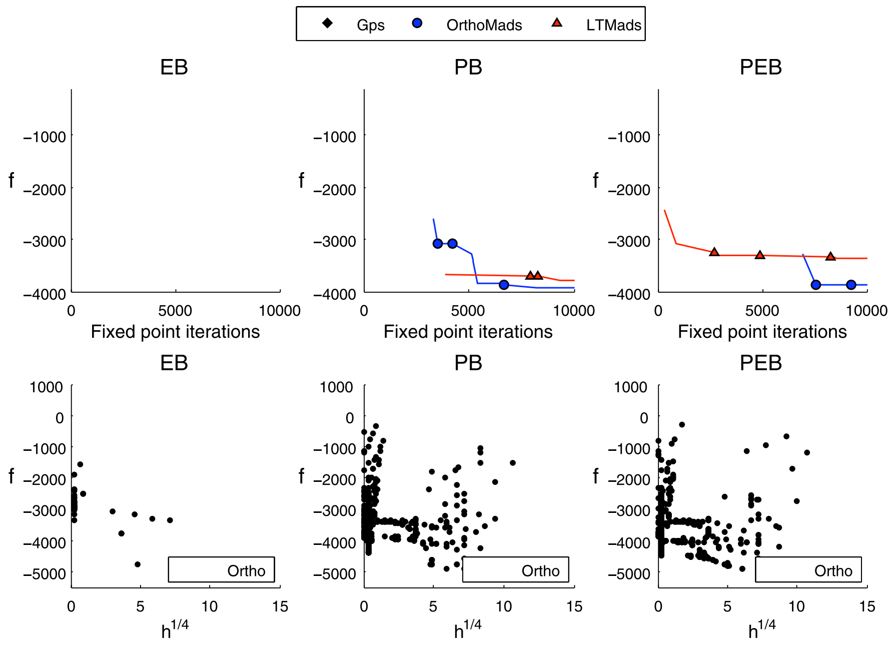

Fig. 5 MDO problem from an infeasible starting point

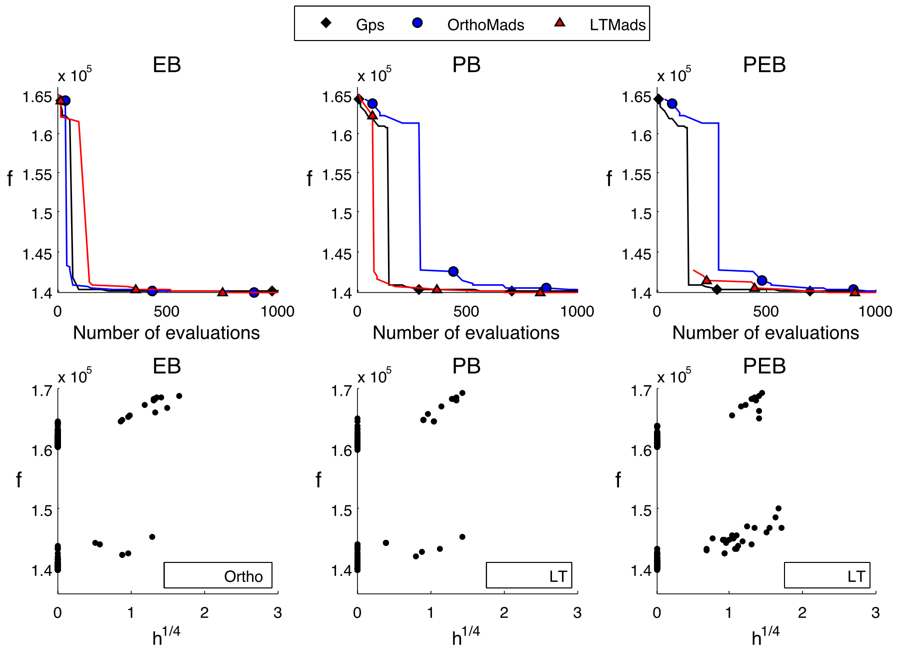

Fig. 6 Well problem from an infeasible starting point

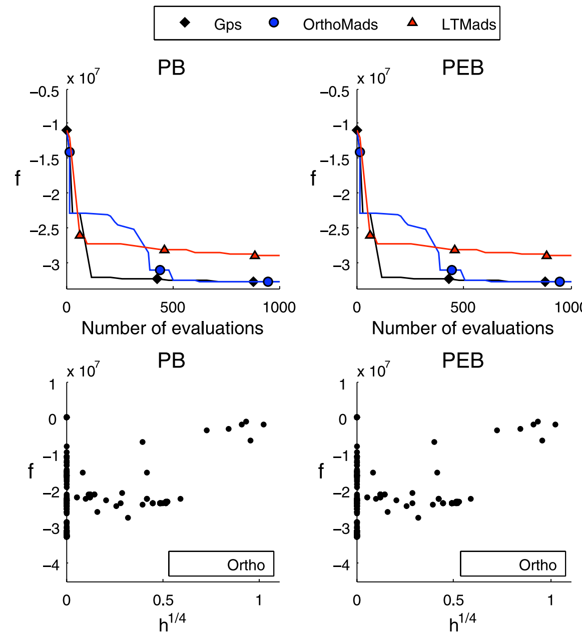

Fig. 7 Styrene problem from a feasible and an infeasible starting points

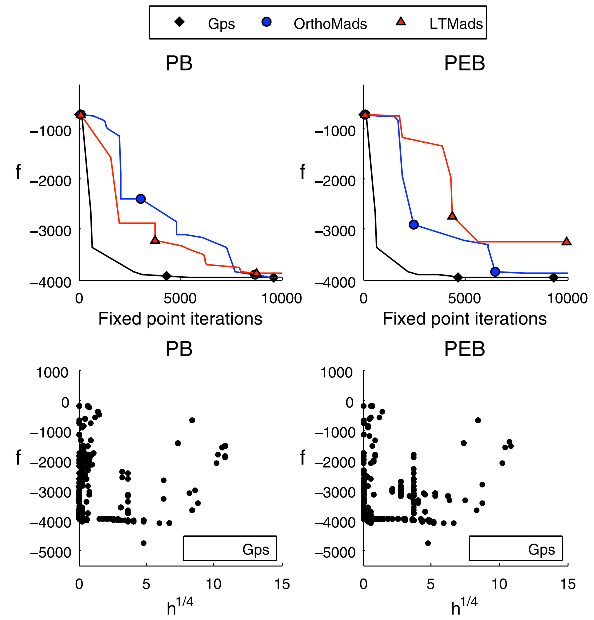

Fig. 8 MDO problem from a feasible and an infeasible starting points

the only GPS variant that did well with either a feasible or infeasible starting point. With both starting points, it does well with either the PB or PEB strategy. Again the secondary poll step is the key.

LTMADS-PB did well, but LTMADS-PEB did not. ORTHOMADS-PB did reach the best known solution, but it was slow for the first 5000 fixed point iterations.

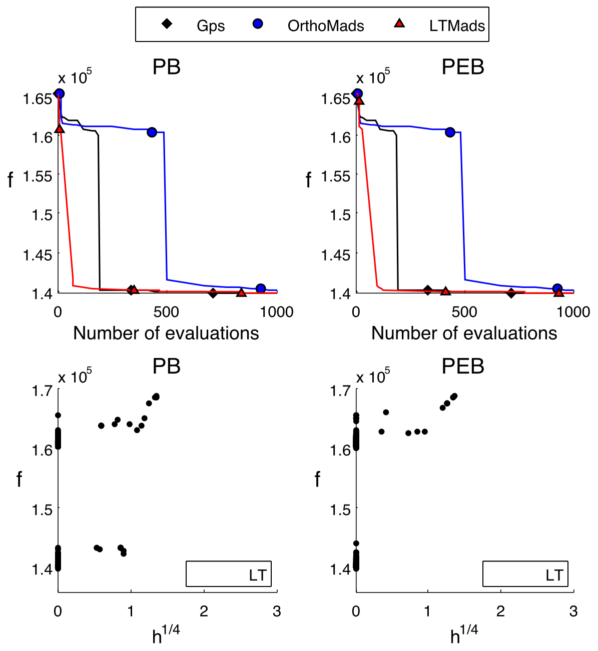

Fig. 9 Well problem from a feasible and an infeasible starting points

Finally, OrthoMads-PEB was faster. We conclude that PB is the best strategy here since all three algorithms use it successfully.

For the last set of tests given in Fig. 9 for the Well problem from two starting points, both PB and PEB have the same behavior. LTMads performed the best, followed by Gps and then by OrthoMads. Gps only uses the coordinate directions, and LTMads uses them more often than OrthoMads. We assume that this is further confirmation of the affinity of this problem for the coordinate directions. However, all methods lead to the best known value.

5.4 Multiple runs of LTMads and OrthoMads

The previous subsections compared single runs of Gps, LTMads and OrthoMads under different constraint handling strategies from different starting points. Another usage of these algorithms consists in launching multiple runs of LTMads or OrthoMads with various random or Halton seeds. Our second phase of tests shows the worst, median and best final solutions for multiple runs. In a multiple run environment, the user is likely to care most about the best solution. However, the runs also indicated the sensitivity of the methods to the randomness in LTMads, and to the controllable parameter Halton seed for OrthoMads. We have already presented results for the default Halton seed.

Table 4 summarizes results of LTMads with 30 different random seeds and OrthoMads with 30 different Halton seeds. For each of the three test problems, each algorithm was executed 60 times under the EB approach, and 90 times under PB as well as PEB. The reason why the EB approach was executed less often is that runs

Table 4 Results for series of several runs of LTMADS and ORTHOMADS with different random seeds and Halton seeds. The symbol ∅ indicates that no feasible solution was found

Problem	EB			PB			PEB		
• Method	Worst	Median	Best	Worst	Median	Best	Worst	Median	Best
	(out of 60 runs)			(out of 90 runs)			(out of 90 runs)		
Styrene									
• LTMADS	−2.89E7	−2.93E7	−3.31E7	−2.60E7	−3.25E7	−3.36E7	−2.60E7	−3.25E7	−3.35E7
• ORTHOMADS	−2.88E7	−2.93E7	−3.31E7	−2.64E7	−3.13E7	−3.32E7	−2.64E7	−3.08E7	−3.32E7
MDO									
• LTMADS	∅	−1754.7	−3964.1	∅	−3871.7	−3963.6	∅	−3664.6	−3962.9
• ORTHOMADS	∅	−1385.9	−3964.0	∅	−3929.6	−3963.6	∅	−3890.9	−3964.1
Well									
• LTMADS	140226	139887	139873	140335	139908	139873	140335	139895	139873
• ORTHOMADS	160287	139890	139873	160287	139971	139873	160287	139962	139873

from both starting points under the extreme barrier were not conducted since they are redundant with those from the feasible starting point.

Analysis of the logs of the runs lead to the following observations. There is little difference in the best solutions, though there is some. Of the nine possible combinations of methods with ways to handle constraints, ORTHOMADS found a better solution than LTMADS only on the MDO problem with the PEB. LTMADS found a better solution than ORTHOMADS on the MDO problem with the EB strategy, and with the PB and PEB on the Styrene problem. The same best solution was found five times, with all three strategies on the Well problem, with PB on the MDO problem, and with the EB on the Styrene problem.

The variations in worst, median, best show that LTMADS is sensitive to the randomness it uses, and ORTHOMADS is sensitive to the Halton seed.

For the Styrene problem, the least sensitive strategy appears to be LTMADS-PB. The median and best values dominate those of EB and PEB for both LTMADS and ORTHOMADS. For the MDO problem, the best strategy appears to be ORTHOMADS with all three strategies.

For the Well problem, the best run of every combination reached the best known solution. However, LTMADS is less sensitive with respect to the random seed used than ORTHOMADS is with respect to the Halton seed for this problem since each of the 240 LTMADS runs produced a solution with an objective function value close to the best known value. Some ORTHOMADS runs were unsuccessful in closing a well, and ended up with a high objective function value. The best strategy for the Well problem appears to be LTMADS-EB. The worst, median and best values beat those of PB and PEB.

In summary, LTMADS appears to be slightly superior to ORTHOMADS when performing multiple runs, which might be useful information in a massively parallel computing environment.

Table 5 Summary of best solutions

Problem	Solution x^*	$f(x^*)$
Styrene	LTMads-PB from infeasible point (unreproducible) (99.53125, 79.224609375, 96.46875, 0, 0, 46.90625, 32.2578125, 48.75)	−33618600
	OrthoMads-PB and OrthoMads-PEB from both starting points (reproducible, Halton seed: 20) (100, 45.0625, 66.5625, 0, 0, 31.3125, 32.7861328125, 49.6875)	−33235500
MDO	OrthoMads-PEB from both starting points (reproducible, Halton seed: 46) Gps-PEB from both starting points (reproducible) LTMads-EB from feasible point (unreproducible) (0.399998626708984, 0.75000114440918, 0.75, 0.156244850158691, 0.059999664306641, 59999.942779541015625, 1.4, 2.500032043457031, 70, 1500)	−3964.14
Well	Gps-EB from infeasible point (reproducible) LTMads for best run of every combination (unreproducible) OrthoMads for best run of every combination (reproducible, e.g., OrthoMads-EB with default Halton seed from feasible point) except PB and PEB from the infeasible point (10, 23, 10, 7, 10, 40, 27, 40, 43, 40, 10, 1, −0.006399598297772, −0.006399599899869, −0.006399599849803, −0.006399598948624, −0.006399599849803)	139873.5

5.5 Best solutions we found by any approach

To aid other researchers who might want to test their own algorithms using these problems, we give Table 5, which displays the overall best solution produced in our tests, including the runs using Gps. It also lists which method (or methods) generated them. When the solution was generated by LTMads, we also included the best reproducible solution.

6 Discussion

It is not surprising that there does not seem to be a combination of the ways to handle constraints with a given algorithmic strategy that is dominant over all others. Though it can be argued that PB is the most reliable of the three strategies for handling constraints. Nonetheless, the 1464 runs summarized in the previous section lead to the following observations.

When launching multiple runs, there appeared to be a slight preference for the LTMads strategy over OrthoMads and for PB over PEB. However, for costly simulations, a user will usually choose one of these algorithms with the default parameters, and launch it only once. In these tests, GPS and LTMADS generally do well, but they are more erratic than ORTHOMADS, which found solutions that were not significantly bested on any of the problems.

The default choices in our C++ software NOMAD [1] are set to ORTHOMADS-PB. The reason why we chose PB over EB or PEB can be seen in Figs. 4, 5 and 8. In these three figures, ORTHOMADS-PB outperformed both ORTHOMADS-EB and ORTHOMADS-PEB. In all other figures, ORTHOMADS-PB was never dominated by the others.

Some might question our choice of C++ for NOMAD based on the prevalence of Fortran in engineering computation. Notice first that the simplicity of our methods requires no significant computation in the body of NOMAD. Instead, the user supplied black-box function codes overwhelmingly carry the computational load. These frequently are given in Fortran, and their interface to NOMAD is easy and well documented. So, we do not lose efficiency at runtime by our choice. On the positive side, using an object oriented language was the right choice because of the importance to our user base of easy customization of NOMAD. This is particularly true for user-supplied surrogate packages and the search routines that employ them.

A referee suggested that we might provide results from a finite-difference gradient-based method as a baseline for these methods. We have not done so, and some explanation is in order. First, we are not really in competition with such methods. When it is worthwhile for the user to take the time to set them up with the routines these more delicate algorithms need, and when the number of decision variables is larger than considered here, then by all means they should be used because they will probably be more efficient when they work. We are interested in problems for which they are unlikely to work right out of the box. Second, although finite-differences can be used to estimate gradients in the presence of hidden and unrelaxable constraints, this is a ticklish business requiring even more than usual care in choosing the step-size. Third is that we are aiming at a different class of problem for which there is not enough precision in the governing simulations to provide useful finite-difference gradients, even without the issue of hidden constraints.

Acknowledgements We are pleased to have this paper appear in an issue celebrating the 60th birthday of our friend and colleague Mario Martinez.

References

1. Abramson, M.A., Audet, C., Couture, G., Dennis, J.E. Jr., Le Digabel, S.: The NOMAD project. Software available at http://www.gerad.ca/nomad
2. Abramson, M.A., Audet, C., Dennis, J.E. Jr., Le Digabel, S.: OrthoMADS: A deterministic MADS instance with orthogonal directions. SIAM J. Optim. (2009, to appear)
3. Audet, C., Dennis, J.E. Jr.: Mesh adaptive direct search algorithms for constrained optimization. SIAM J. Optim. **17**(1), 188–217 (2006)
4. Audet, C., Dennis, J.E. Jr.: A progressive barrier for derivative-free nonlinear programming. SIAM J. Optim. **20**(4), 445–472 (2009)
5. 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)

6. Booker, A.J., Dennis, J.E. Jr., Frank, P.D., Serafini, D.B., Torczon, V.: Optimization using surrogate objectives on a helicopter test example. In: Borggaard, J., Burns, J., Cliff, E., Schreck, S. (eds.) Optimal Design and Control, Progress in Systems and Control Theory, pp. 49–58. Birkhäuser, Cambridge (1998)
7. Booker, A.J., Dennis, J.E. Jr., Frank, P.D., Serafini, D.B., Torczon, V., Trosset, M.W.: A rigorous framework for optimization of expensive functions by surrogates. Struct. Optim. **17**(1), 1–13 (1999)
8. Choi, T.D., Kelley, C.T.: Superlinear convergence and implicit filtering. SIAM J. Optim. **10**(4), 1149–1162 (2000)
9. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-Free Optimization. MPS/SIAM Book Series on Optimization. SIAM, Philadelphia (2009)
10. Fiacco, A.V., McCormick, G.P.: Nonlinear Programming: Sequential Unconstrained Minimization Techniques. Wiley, New York (1968). Reissued in 1990 by SIAM Publications, Philadelphia, as Vol. 4 in the series Classics in Applied Mathematics
11. Fletcher, R., Leyffer, S.: Nonlinear programming without a penalty function. Math. Program. Ser. A **91**, 239–269 (2002)
12. Fletcher, R., Leyffer, S., Toint, Ph.L.: On the global convergence of a filter–SQP algorithm. SIAM J. Optim. **13**(1), 44–59 (2002)
13. Fletcher, R., Leyffer, S., Toint, Ph.L.: A brief history of filter methods. SIAM SIAG/OPT Views-and-News **18**(1), 2–12 (2006)
14. Fowler, K.R., Reese, J.P., Kees, C.E., Dennis, J.E. Jr., Kelley, C.T., Miller, C.T., Audet, C., Booker, A.J., Couture, G., Darwin, R.W., Farthing, M.W., Finkel, D.E., Gablonsky, J.M., Gray, G., Kolda, T.G.: Comparison of derivative-free optimization methods for groundwater supply and hydraulic capture community problems. Adv. Water Resour. **31**(5), 743–757 (2008)
15. Gill, P.E., Murray, W., Saunders, M.A.: User's guide for SNOPT version 7: Software for large-scale nonlinear programming (2006)
16. Halton, J.H.: On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numer. Math. **2**(1), 84–90 (1960)
17. Marsden, A.L.: Aerodynamic noise control by optimal shape design. Ph.D. thesis, Stanford University (2004)
18. 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
19. Sobieszczanski-Sobieski, J., Agte, J.S., Sandusky, R.R. Jr.: Bi-level integrated system synthesis (BLISS). Technical Report NASA/TM-1998-208715, NASA, Langley Research Center, August 1998
20. Torczon, V.: On the convergence of pattern search algorithms. SIAM J. Optim. **7**, 1–25 (1997)
21. Waltz, R.A., Plantenga, T.D.: KNITRO user's manual, version 6.0. Ziena optimization, inc. (2009)