Les Cahiers du GERAD ISSN: 0711–2440

Efficient solution of quadratically constrained quadratic subproblems within a direct-search algorithm

N. Amaioua, C. Audet,
A. R. Conn, S. Le Digabel

G–2016–45

June 2016
Revised: November 2016

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Efficient solution of quadratically constrained quadratic subproblems within a direct-search algorithm

Nadir Amaioua [a]
Charles Audet [a]
Andrew R. Conn [b]
Sébastien Le Digabel [a]

[a] *GERAD and Département de mathématiques et génie industriel, École Polytechnique de Montréal, Montréal (Québec), Canada H3C 3A7*

[b] *IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, États-Unis*

nadir.amaioua@gerad.ca
charles.audet@gerad.ca
arconn@us.ibm.com
sebastien.le.digabel@gerad.ca

June 2016
Revised: November 2016

Les Cahiers du GERAD
G–2016–45

Abstract: The Mesh Adaptive Direct Search algorithm (MADS) is an iterative method for constrained blackbox optimization problems. One of the optional MADS features is a versatile search step in which quadratic models are built leading to a series of quadratically constrained quadratic subproblems. This work explores different algorithms that exploit the structure of the quadratic models: the first one applies an l_1 exact penalty function, the second uses an augmented Lagrangian and the third one combines the former two, resulting in a new algorithm. These methods are implemented within the NOMAD software package and their impact are assessed through computational experiments on 65 analytical test problems and 4 simulation-based engineering applications.

Keywords: Derivative-free optimization, quadratic programming, trust-region subproblem, Mesh Adaptive Direct Search

Acknowledgments: This work is supported by the NSERC CRD RDCPJ 490744-15 grant in collaboration with Hydro-Québec and Rio Tinto.

1 Introduction

We consider the following constrained optimization problem:

$$\begin{array}{ll} \min\limits_{x\in\mathcal{X}} & f(x) \\ \text{subject to} & c_j(x) \leq 0, \quad j \in [\![1,m]\!] \end{array} \tag{1}$$

where m is a positive integer, $\mathcal{X}$ is a subset of $\mathbb{R}^n$, f and $(c_j)_{j\in[\![1,m]\!]}$ are real-valued functions, possibly evaluated by a computer simulation, seen as a blackbox with the following characteristics: function evaluations take a long time to compute, the simulation may fail for some input values, the derivatives are not available and their approximations may be unreliable. The set $\mathcal{X}$ is often defined by bound on the variables.

Derivative-free optimization (DFO, [19]) algorithms are designed to handle this type of problems. DFO methods do not use or try to approximate derivatives of the problems. Instead, they either rely on a direct search approach which uses a discretization of the solution space and generates directions to test trial points, or they use model-based methods by constructing polynomial approximations to mimic the functions over some specified trust-region.

The present work focuses on the Mesh Adaptive Direct Search algorithm (MADS) [4] with quadratic models [18]. MADS principally relies on a pair of steps, called the search and the poll, to explore the space of variables and a third step to update its parameters. Both the search and the poll are complementary: the search allows local and global exploration while the poll is local and ensures convergence. We consider a model-based approach in the search step that has no impact on the theoretical convergence analysis of MADS, but improves its practical performance. At each iteration k, the search builds local quadratic models near the current iterate x^k for the objective function f and for each of the m inequality constraints. This leads to a quadratically constrained quadratic subproblem over an infinity norm trust-region:

$$\begin{array}{lll} \min\limits_{x\in\mathcal{X}} & f^k(x) & \\ \text{subject to} & c_j^k \leq 0, & j \in [\![1,m]\!] \\ & \left\|x - x^k\right\|_\infty \leq \Delta^k & \end{array} \tag{2}$$

where the scalar $\Delta^k \geq 0$ defines the trust-region, f^k is the quadratic model of the objective function near the current iterate x^k and the $(c_j^k)_{j\in[\![1,m]\!]}$ are the quadratic models of the m inequality constraints.

In [18], to solve Problem (1), a series of subproblems of the type (2) are created and each of them is solved with a new instance of MADS. The paper concludes by stating that the quadratic structure of the subproblems is not exploited and that a blackbox optimization solver is certainly not the appropriate choice to solve a quadratically constrained quadratic problem. The main purpose of this paper is to test dedicated algorithms to solve Problem (2). We choose two widely known methods from the literature: the l_1 exact penalty function and the augmented Lagrangian methods. A new approach combining the strengths of both methods is introduced and called the l_1 augmented Lagrangian: it uses an l_1 penalty term instead of the squared one used in the standard augmented Lagrangian.

This paper is organized as follows. Section 2 describes the MADS algorithm and in particular, its mechanisms to build and use quadratic models. The section also summarizes several methods from the literature handling quadratic subproblems. Section 3 presents the l_1 exact penalty function algorithm and Section 4 gives a description of the augmented Lagrangian method. Section 5 introduces the new l_1 augmented Lagrangian method and describes implementation choices. Section 6 compares the results of the three algorithms on a set of 61 analytical and 4 simulation-based test problems and Section 7 concludes with recommendations.

2 Literature review

The MADS algorithm is an iterative method that uses a discretization of the solution space called the mesh to select and evaluate new trial points. Each iteration of MADS consists of two main steps: the search and poll, followed by a parameter update step.

The poll and update steps are critical to the convergence proof of MADS [4], but the search is optional and more flexible: it can be omitted or defined by the user. A frequently used search strategy [18] is to automatically construct quadratic models to try and find a promising trial point. If the search succeeds, i.e. the selected trial point improves upon the current iterate, then this trial point becomes the new iterate and the poll step is skipped. On the contrary, if the search fails, the poll step becomes mandatory. Other types of search step such as VNS [3] and surrogate-based [7, 9] are not the topic of the present paper.

The poll is used to choose mesh points near the current iterate and to evaluate their objective and constraint values. On the one hand, if the poll fails to find a better solution, the update step will reduce the mesh size (the parameter that scales the space discretization) and the poll size (maximum distance allowed between a trial point and the current iterate) in order to concentrate near the current iterate. On the other hand, once a better solution is found, the poll step terminates and the update step increases the mesh size. The diagram in Figure 1 represents a description of the MADS algorithm with a search step based on quadratic models.

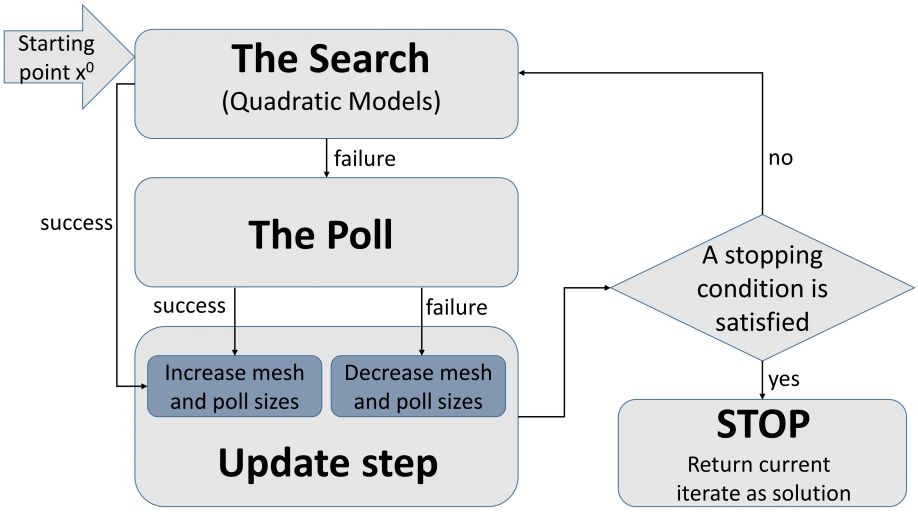


Figure 1: An overview of the MADS algorithm with a search step based on quadratic models.

The quadratic models constructed in the search step are also used in the poll step: the poll produces a list of trial mesh points, and instead of sending them directly to be evaluated, quadratic models of the objective function and of the constraints are used to sort the trial points, so that the most promising ones are evaluated first. This approach is used in conjunction with the opportunistic strategy: as soon as a better solution than the current iterate is found, the poll step stops without executing the simulation at the remaining trial points. Using quadratic models in both the search and poll steps greatly improves the MADS performance, as reported in [18].

At each iteration of MADS, a quadratic subproblem of the form (2) is constructed and solved within the search step. The resulting solution is projected on the mesh to provide a starting point that satisfies the convergence requirements of MADS. The subproblem is similar to those arising in trust-region methods called trust-region subproblems. Since the early eighties, many algorithms have been developed specifically to solve this kind of quadratic problems. The Moré & Sorenson algorithm (MS) [40] is one of the first algorithms used specifically for quadratic problems subject to an ellipsoidal constraint. It is based on solving the optimality conditions via the Newton algorithm with backtracking. The downside of this algorithm is that it uses

Cholesky factorizations that become expensive for large matrices. Later, the Generalized Lanczos Trust-Region algorithm (GLTR) [23] was implemented by using the MS algorithm on Krylov subspaces. However, GLTR could not handle hard cases (see chapter 7 in [17]) of the trust-region subproblems. The same principle is applied by the Sequential Subspace Method (SSM) [28] that creates four dimensional subspaces instead of using the Krylov subspaces. Even if the SSM algorithm handles the hard case of the trust-region subproblem, it still solves quadratic problems over a sphere. The Gould-Robinson & Thorne algorithm [25] improved the MS algorithm by using some high dimensional polynomial approximations that allow the Newton method to converge in fewer iterations. Another algorithm, that treats specifically quadratic problems over a convex quadratic constraint, is the Rendl & Wolkowicz algorithm [21, 46] which rewrites the problem into an eigenvalue subproblem that can be handled by the Newton method combined with Armijo-Goldstein conditions. This algorithm is suitable for large-scale matrix problems.

All the algorithms above are used for a quadratic objective over a unique constraint defining the trust-region. In our case, Problem (2) is additionally constrained by m quadratic constraints and, recently, one method was developed specifically for this kind of problems using an extension of the Rendl & Wolkowicz algorithm [44]. Solving Problem (2) can also be done by nonlinear optimization tools such as exact penalty functions and augmented Lagrangians. These two approaches are discussed further in the next two sections.

3 The l_1 exact penalty function (l_1EPF algorithm)

The l_1 exact penalty function starts by transforming Problem (2) into the following bound-constrained problem:

$$\begin{array}{ll} \min\limits_{x\in\mathcal{X}} & f^k(x) + \frac{1}{\mu}\sum\limits_{j=1}^{m} \max(0, c_j^k(x)) \\ \text{subject to} & \left\|x - x^k\right\|_\infty \leq \Delta^k \end{array} \tag{3}$$

where $\mu \in \mathbb{R}^+$ is the penalty coefficient. This method was introduced by Zangwill in [52] and Pietrzykowski showed in [49] that, for μ sufficiently small, if the objective function and the constraints are continuously differentiable, any local minimum of a nonlinear problem would also be a minimizer of the l_1 penalty function of that problem. In [35] and [52], it is shown that, in the convex case, there exists a threshold μ_0 such as when μ is smaller than μ_0, a minimizer of the l_1 penalty function would also be a minimizer of the nonlinear problem. Charalambous [10] generalizes this results under second order conditions. In [11], Coleman and Conn explore optimality conditions of the l_1 penalty function. For our implementation, we concentrate on the approach provided by Coleman and Conn in [12, 13].

The objective function of (3) is not differentiable on the boundary of the domain defined by $c^k(x) \leq 0$. We introduce two subsets of indices at some point $\tilde{x} \in \mathcal{X}$:

$$A_\epsilon = \left\{j \in [\![1, m]\!] : |c_j^k(\tilde{x})| \leq \epsilon\right\} \quad \text{and} \quad V_\epsilon = \left\{j \in [\![1, m]\!] : \ c_j^k(\tilde{x}) \ > \epsilon\right\}. \tag{4}$$

The set A_ϵ contains the indices of ϵ-active constraints at $\tilde{x}$, and V_ϵ regroups all the indices of the ϵ-violated constraints at $\tilde{x}$. The remaining constraints whose indices are in $\{j \in [\![1, m]\!] : c_j^k(\tilde{x}) \leq -\epsilon\}$ (ϵ-satisfied) do not contribute locally to the objective function due to the l_1 term. Thus, for some neighborhood $\mathcal{N}(\tilde{x}) \subseteq \mathcal{X}$ of $\tilde{x}$, Problem (3) may be rewritten as:

$$\begin{array}{ll} \min\limits_{x\in\mathcal{N}(\tilde{x})} & p^k(x,\mu) + \frac{1}{\mu}\sum\limits_{j\in A_\epsilon} \max(0, c_j^k(x)) \\ \text{subject to} & \left\|x - x^k\right\|_\infty \leq \Delta^k \end{array} \tag{5}$$

where the penalty function $p^k(x,\mu) := f^k(x) + \frac{1}{\mu}\sum\limits_{j\in V_\epsilon} \max(0, c_j^k(x))$ is differentiable at $x \in \mathcal{N}(\tilde{x})$. The second part of the objective function of (5) is not locally differentiable. In order to overcome this difficulty, we would

like to find a descent direction $h \in \mathbb{R}^n$ that would prevent any change in the ϵ-active constraints up to the second order while minimizing the variation in the penalty function:

$$\begin{array}{ll} \min\limits_{h\in\mathbb{R}^n} & \nabla_x p^k(\tilde{x},\mu)^\top h + \frac{1}{2} h^\top \nabla_{xx} p^k(\tilde{x},\mu) h \\ \text{subject to} & \nabla c_j^k(\tilde{x})^\top h + \frac{1}{2} h^\top \nabla^2 c_j^k(\tilde{x}) h = 0, \qquad j \in A_\epsilon. \end{array} \tag{6}$$

Solving (6) approximately is detailed by Coleman and Conn in [13] and this process circumvents the differentiability issues and explores a subspace where all the constraints are far from being satisfied. We call h a range space minimization direction and it is used to reduce the penalty function while keeping the ϵ-active constraints constant. That is why each time $\tilde{x} + h$ is close to a point x^* that satisfies second-order sufficiency conditions, a null space minimization direction v is also needed and is used to improve the action of the ϵ-active constraints:

$$\phi_{A_\epsilon}^k(\tilde{x} + h + v) = 0 \tag{7}$$

with $\phi_{A_\epsilon}^k(x) = [c_j^k(x)]_{j\in A_\epsilon}$ the vector of ϵ-active constraints values at x. Using a Taylor expansion up to the first order, the expression becomes:

$$\phi_{A_\epsilon}^k(\tilde{x} + h) + J_{A_\epsilon}^k(\tilde{x})^\top v \approx 0 \tag{8}$$

with $J_{A_\epsilon}^k(x) = [\nabla c_j^k(x)]_{j\in A_\epsilon}$ the matrix of ϵ-active constraints gradients. Then v can be computed by solving the linear system:

$$J_{A_\epsilon}^k(\tilde{x})^\top v = -\phi_{A_\epsilon}^k(\tilde{x} + h). \tag{9}$$

When v is computed, the iterate is updated along the direction $h+v$ with a stepsize of 1 only if it presents a sufficient decrease. Algorithm 1 summarizes the inner iteration of the l_1EPF algorithm.

Algorithm 1 l_1EPF algorithm (inner iteration) [13]

1: Initialization of ϵ, x, A_ϵ and V_ϵ
2: If the stopping criteria are met: STOP
   Otherwise go to 3
3: Compute h (by approximately solving (6))
4: **if** x is close to a point satisfying the 2^{nd} order sufficiency conditions **then**
5: Compute v (by solving (9))
6: **if** $x + h + v$ decreases p sufficiently **then**
7: $x \leftarrow x + h + v$
8: **else**
9: Update ϵ, A_ϵ and V_ϵ
10: Compute h (by approximately solving (6))
11: Compute the step size β (Line-search algorithm)
12: $x \leftarrow x + \beta h$
13: **end if**
14: **else**
15: Compute the step size β (Line-search algorithm)
16: $x \leftarrow x + \beta h$
17: **end if**
18: Go to 2.

Algorithm 1 provides an overview of the methodology used to select a descent direction (See [13] for the complete description). A line-search algorithm tailored for piecewise differentiable functions is described in [13]. The overall algorithm tries simply, at each iteration, to find a minimizer of the penalty function for a fixed value of μ (Algorithm 1) and then proceeds to reduce μ if the selected minimizer is infeasible for (2) in order to put more weight on the constraints. Algorithm 2 describes the outer iteration of the l_1EPF method.

Algorithm 2 l_1EPF algorithm (outer iteration) [13]

1: Initialization of μ and x
2: **while** x is not feasible **do**
3: Minimize p to find new iterate x (Algorithm 1)
4: Update: $\mu = \mu/10$
5: **end while**

4 Augmented Lagrangian (AugLag)

The augmented Lagrangian method was introduced by Powell [45] and Hestenes [30] in the late sixties. Since that time, the method was widely studied, improved and supported by convergence analyses [27, 43] and different versions of the algorithm were implemented. A well-known implementation is the LANCELOT package [14, 16]. When using slack variables in the implementation, it is possible to exploit the structure of these variables to increase the efficiency of the algorithm [15, 20]. An augmented Lagrangian for constrained blackbox problems is designed in [26]. The paper [8] introduces a derivative-free trust-region algorithm (DFTR) that uses the augmented Lagrangian approach to solve quadratically constrained quadratic subproblems and provides a method to update the trust-region radius. In the present work, we follow the approach described in [2] which is similar to the LANCELOT implementation.

First, slack variables $s \in \mathbb{R}^m$ are added to Problem (2):

$$\begin{array}{ll} \min\limits_{x\in\mathcal{X},\ s\in\mathbb{R}^m} & f^k(x) \\ \text{subject to} & c_j^k(x) + s_j = 0, \qquad j \in [\![1, m]\!] \\ & \left\|x - x^k\right\|_\infty \leq \Delta^k, \\ & s_j \geq 0, \qquad j \in [\![1, m]\!] \end{array} \tag{10}$$

and the bound constrained augmented Lagrangian is defined as follows:

$$\begin{array}{lll} \min\limits_{x\in\mathcal{X},\ s\in\mathbb{R}^m} & \mathcal{L}_a^k(x, s, \lambda, \mu) & = f^k(x) - \sum\limits_{j=1}^{m} \lambda_j(c_j^k(x) + s_j) + \frac{1}{2\mu}\sum\limits_{j=1}^{m}(c_j^k(x) + s_j)^2 \\ \text{subject to} & \left\|x - x^k\right\|_\infty & \leq \Delta^k, \\ & s_j \geq 0, & j \in [\![1, m]\!] \end{array} \tag{11}$$

where $\lambda \in \mathbb{R}^m$ and $\mu \in \mathbb{R}^+$. The augmented Lagrangian algorithm solves (11) at each iteration l of the outer algorithm for fixed values λ^l and μ^l to compute the next iterate (x^{l+1}, s^{l+1}). The index l counts the outer iterations of the augmented Lagrangian method while the index k characterizes the subproblem (2). The outer iteration proceeds to update one of the parameters λ^l or μ^l: when all the constraints values are below a given tolerance value, η^l, the iterate is judged locally feasible and the algorithm proceeds to update the multipliers λ^l. In the other case, the penalty parameter μ is reduced in order to give more importance to the constraints and try to satisfy them on the next iteration. Algorithm 3 gives an overview of the outer iteration of the augmented Lagrangian method:

To compute the next iterate in step 3 of Algorithm 3, a quadratic model is used to approximate (11) at each iteration l of the outer loop:

$$\begin{array}{ll} \min\limits_{d^x\in\mathbb{R}^n,\ d^s\in\mathbb{R}^m} & \nabla_x\mathcal{L}_a^k(x^l, s^l, \lambda^l, \mu^l)^\top d^x + \frac{1}{2}d^{x\top}\nabla_{xx}\mathcal{L}_a^k(x^l, s^l, \lambda^l, \mu^l)d^x + \\ & \nabla_s\mathcal{L}_a^k(x^l, s^l, \lambda^l, \mu^l)^\top d^s + \frac{1}{2}d^{s\top}\nabla_{ss}\mathcal{L}_a^k(x^l, s^l, \lambda^l, \mu^l)d^s \\ \text{subject to} & \left\|x^l + d^x - x^k\right\|_\infty \leq \Delta^k, \\ & s^l + d^s \geq 0, \\ & \|d^x\|_\infty \leq \delta^l, \\ & \|d^s\|_\infty \leq \delta^l \end{array} \tag{12}$$

Algorithm 3 Augmented Lagrangian to solve Problem (10): outer iteration [2]

1: Initializations: $l = 0,\ \lambda^l,\ \mu^l,\ \eta^l,\ x^l,\ s^l$
2: If the stopping criteria are met: STOP
Otherwise go to 3
3: Compute the next iterate: (x^{l+1}, s^{l+1}) by solving (11)
for fixed values of λ^l and μ^l (Algorithm 5)
4: **if** $\left\|c(x^{l+1}) + s^{l+1}\right\|_\infty \leq \eta^l$ **then**
5: $\quad \lambda^{l+1} \leftarrow \lambda^l - \frac{1}{\mu^l}(c^k(x^l) + s^l)$
6: $\quad \eta^{l+1} \leftarrow \eta^l(\mu^l)^{0.9}$
7: **else**
8: $\quad \mu_{l+1} \leftarrow \frac{\mu_l}{10}$
9: $\quad \eta^{l+1} \leftarrow \frac{(\mu^{l+1})^{0.1}}{10}$
10: **end if**
11: $l \leftarrow l + 1$
12: Go to 2.

with δ^l the trust-region of the model. To simplify, let $d \in \mathbb{R}^{n+m}$ be the concatenation of d^x and d^s ($d^\top = [d^{x\top} d^{s\top}]$) and $y \in \mathbb{R}^{n+m}$ the concatenation of x and s ($y^\top = [x^\top s^\top]$). Since the constraints of (12) form a box $\mathcal{B}$, it is possible to define lower and upper bounds $l \in \mathbb{R}^{n+m}$ and $u \in \mathbb{R}^{n+m}$ and define the feasible set:

$$\mathcal{B} = \quad \{d \in \mathbb{R}^{n+m} : l \leq d \leq u\}. \tag{13}$$

Problem (12) is rewritten as:

$$\min_{d \in \mathcal{B}} \quad \nabla_y \mathcal{L}_a^k(y^l, \lambda^l, \mu^l)^\top d + \tfrac{1}{2} d^\top \nabla_{yy} \mathcal{L}_a^k(y^l, \lambda^l, \mu^l) d. \tag{14}$$

Solving (14) relies on an iterative method developed by Moré & Toraldo to treat box constrained quadratic problems (see [2, 41]). We chose this approach due to some preliminary results on unconstrained problems that favored this trust-region method over the line-search method used in the previous section. The Moré & Toraldo algorithm manipulates the active-set $\mathcal{A}(\tilde{d}) = \{i \in [\![1, n+m]\!] : \tilde{d}_i = \ell_i \text{ or } \tilde{d}_i = u_i\}$ to look for a descent direction p that would be used to update its current iterate $\tilde{d}$ via a line-search. In the first stage of the algorithm, a promising face containing $\tilde{d}$ is selected via a projected gradient method. A face of $\mathcal{B}$ is defined as a set of vectors for which active-set $\mathcal{A}(\tilde{d})$ coordinates coincide with $\tilde{d}$:

$$\mathcal{F} = \quad \{w \in \mathcal{B} : w_i = \tilde{d}_i \text{ for } i \in \mathcal{A}(\tilde{d})\}. \tag{15}$$

The second part of the Moré & Toraldo method concentrates on finding a descent direction p on the selected face with a conjugate gradient algorithm and update $\tilde{d}$. Algorithm 4 summarizes the Moré & Toraldo method to compute the solution d to (14):

Algorithm 4 algorithm PG/CG [41]

1: **while** no solution d is found **do**
2: $\quad$ Use the projected gradient (PG) method to select a promising face of the feasible set $\mathcal{B}$.
3: $\quad$ Use the conjugate gradient (CG) method to explore the selected face and find a descent direction p.
4: $\quad$ Use a projected line-search alongside p to choose the next iterate d.
5: **end while**

Once a direction d is computed, the inner iteration of the augmented Lagrangian method uses a typical trust-region approach: the ratio r (defined in Step 4 of Algorithm 5) determines whether the model fits the augmented Lagrangian function well. If the ratio is close to 1, the model fits the function well and, in that case, the trust-region radius must increase. On the contrary, if the ratio is negative or close to zero, the trust-region must be reduced. The ratio r also allows one to decide if the iterate x gets updated or not. Algorithm 5 describes the inner iteration of the augmented Lagrangian method.

Algorithm 5 Augmented Lagrangian: inner iteration [2]

1: Initializations: $0 < \epsilon_1 \leq \epsilon_2 < 1$, d and $0 < \gamma_1 < 1 < \gamma_2$
2: If the stopping criteria are met: STOP
Otherwise go to 3
3: Solve (14) to find d
4: Compute the ratio : $r = \dfrac{\mathcal{L}_a^k(y^l + d, \lambda^l, \mu^l) - \mathcal{L}_a^k(y^l, \lambda^l, \mu^l)}{\nabla \mathcal{L}_a^k(y^l, \lambda^l, \mu^l)^\top d + \frac{1}{2} d^\top \nabla^2 \mathcal{L}_a^k(y^l, \lambda^l, \mu^l) d}$
5: **if** $r \geq \epsilon_1$ **then**
6: $x^l \leftarrow x^l + d$
7: **if** $r \geq \epsilon_2$ **then**
8: $\delta \leftarrow \gamma_2 \max(\|d\|_\infty, \delta)$
9: **end if**
10: **else**
11: $\delta \leftarrow \gamma_1 \|d\|_\infty$
12: **end if**
13: Go to 2.

5 Augmented Lagrangian with l_1 penalty term (l_1AugLag)

The present section proposes a new method that combines the algorithms from the previous two sections. Since the constraints in Problem (2) are quadratic rather than general nonlinear, the augmented Lagrangian loses the quadratic structure of the subproblem due to the squared penalty term. The idea is to replace this part with a l_1 penalty term in order to get a piecewise quadratic function. We define the l_1 augmented Lagrangian problem in the following manner:

$$\begin{array}{ll} \min\limits_{x \in \mathcal{X}} & f^k(x) - \sum\limits_{j=1}^{m} \lambda_j c_j^k(x) + \frac{1}{\mu} \sum\limits_{j=1}^{m} \max\{0, c_j^k(x)\} \\ \text{subject to} & \left\|x - x^k\right\|_\infty \leq \Delta^k. \end{array} \tag{16}$$

By using the same definition as in Equation (4) of ϵ-active and ϵ-violated sets for a point $\tilde{x} = x^l$ (at iteration l of the outer approximation), it is possible to restrict the analysis of Problem (16) in a neighborhood $\mathcal{N}(x^l) \subseteq \mathcal{X}$ of x^l (same approach as in (5)):

$$\begin{array}{ll} \min\limits_{x \in \mathcal{N}(x^l)} & D^k(x, \lambda, \mu) + \frac{1}{\mu} \sum\limits_{j \in A_\epsilon} \max\{0, c_j^k(x)\} \\ \text{subject to} & \left\|x - x^k\right\|_\infty \leq \Delta^k \end{array} \tag{17}$$

where $D^k(x, \lambda, \mu) = f^k(x) - \sum\limits_{j=1}^{m} \lambda_j c_j^k(x) + \frac{1}{\mu} \sum\limits_{j \in V_\epsilon} c_j^k(x)$ is differentiable, but the second term involving A_ϵ is not. This is similar to the l_1 exact penalty approach: At each iteration l, we need to handle the differentiability issues by concentrating on computing a direction that minimizes the variation in the l_1 augmented Lagrangian expression and keeps the ϵ-active constraints from varying for fixed values of λ and μ:

$$\begin{array}{lll} \min\limits_{h \in \mathbb{R}^n} & \nabla_x D^k(x^l, \lambda^l, \mu^l)^\top h + \frac{1}{2} h^\top \nabla_{xx} D^k(x^l, \lambda^l, \mu^l) h & \\ \text{subject to} & \nabla c_j^k(x^l)^\top h + \frac{1}{2} h^\top \nabla^2 c_j^k(x^l) h = 0, & j \in A_\epsilon. \end{array} \tag{18}$$

To further satisfy the ϵ-active constraints, we can use (9) to compute a null space minimization direction v. Algorithm 1 still applies: we just have to compute the range space minimization direction h in Steps 3 and 10 by solving (18) instead. The outer iteration algorithm has to update either λ or μ which is why the l_1 exact penalty approach alone is not preferred. Instead, we can use a method similar to the outer iteration algorithm of the augmented Lagrangian: we can keep updating the penalty parameter μ in the same way, but the λ update needs some adjustment. We can determine how to update λ by using the optimality test:

$$\nabla_x^k \mathcal{L}(x^l, \lambda^l) = 0 \tag{19}$$

with $\mathcal{L}(x,\lambda) = f(x) - \sum_{j=1}^{m} \lambda_j c_j^k(x)$, the Lagrangian of Problem (2). So, at iteration l, if x_{l+1} is a minimizer of (17), then there exists a vector $\bar{\lambda}$ for which:

$$\nabla f^k(x^{l+1}) - \sum_{j=1}^{m} \lambda_j \nabla c_j^k(x^{l+1}) + \frac{1}{\mu} \sum_{j \in V_\epsilon} \nabla c_j^k(x^{l+1}) - \sum_{j \in A_\epsilon} \bar{\lambda}_j \nabla c_j^k(x^{l+1}) = 0. \tag{20}$$

If we define λ^{l+1} to be:

$$\lambda^{l+1} = \begin{cases} \lambda_j^l + \bar{\lambda}_j, & \text{if } j \in A_\epsilon \\ \lambda_j^l - \frac{1}{\mu}, & \text{if } j \in V_\epsilon \\ \lambda_j^l, & \text{if } j \notin A_\epsilon \cup V_\epsilon \end{cases} \tag{21}$$

then the optimality test (19) becomes:

$$\begin{aligned} \nabla_x \mathcal{L}(x^{l+1}, \lambda^{l+1}) &= \nabla f^k(x^{l+1}) - \sum_{j=1}^{m} \lambda_j^{l+1} \nabla c_j^k(x^{l+1}) \\ &= \nabla f^k(x^{l+1}) - \sum_{j \notin A_\epsilon \cup V_\epsilon} \lambda_j^l \nabla c_j^k(x^{l+1}) \\ &\quad - \sum_{j \in V_\epsilon} (\lambda_j^l - \tfrac{1}{\mu}) \nabla c_j^k(x^{l+1}) - \sum_{j \in A_\epsilon} (\lambda_j^l + \bar{\lambda}_j) \nabla c_j^k(x^{l+1}) \\ &= \nabla f^k(x^{l+1}) - \sum_{j=1}^{m} \lambda_j \nabla c_j^k(x^{l+1}) + \tfrac{1}{\mu} \sum_{j \in V_\epsilon} \nabla c_j^k(x^{l+1}) \\ &\quad - \sum_{j \in A_\epsilon} \bar{\lambda}_j \nabla c_j^k(x^{l+1}) \\ &= 0 \end{aligned} \tag{22}$$

which confirms our update method of λ. The only remaining issue is how to compute $\bar{\lambda}$. Equation (20) is rewritten as:

$$\sum_{j \in A_\epsilon} \bar{\lambda}_j \nabla c_j^k(x^{l+1}) = \nabla f^k(x^{l+1}) - \sum_{j=1}^{m} \lambda_j \nabla c_j^k(x^{l+1}) + \frac{1}{\mu} \sum_{j \in V_\epsilon} \nabla c_j^k(x^{l+1}). \tag{23}$$

By using matrix notations, Equation (23) becomes:

$$J_{A_\epsilon}^k \bar{\lambda} = \nabla D^k(x^{l+1}, \lambda^{l+1}, \mu^{l+1}) \tag{24}$$

where $J_{A_\epsilon}^k$ is the matrix of ϵ-active constraint gradients defined in Section 3. By using a QR decomposition, there exists an orthogonal matrix Q and an upper triangular matrix U such that:

$$J_{A_\epsilon}^k = QR = [Q_1 \; Q_2] \begin{bmatrix} U \\ 0 \end{bmatrix} = Q_1 U \tag{25}$$

where the number of columns of Q_1 is the same as the number of rows of U and corresponds to the number of ϵ-active constraints near $\tilde{x}$. We can use this to transform (24) into:

$$U \; \bar{\lambda} = Q_1^\top \nabla D^k(x^{l+1}, \lambda^{l+1}, \mu^{l+1}). \tag{26}$$

It is now easy to get $\bar{\lambda}$ by using a backward substitution algorithm. Algorithm 6 displays the outer iteration of the l_1AUGLAG method:

Algorithm 6 l_1 augmented Lagrangian: outer iteration

1: Initializations: $l = 0,\ \lambda^l,\ \mu^l,\ \eta^l,\ x^l,\ s^l$
2: If the stopping criteria are met: STOP
Otherwise go to 3
3: Compute the next iterate: x^{l+1} by minimizing (17)
for fixed values of λ^l and μ^l.
4: **if** $c_j^k(x^{l+1}) \leq \eta^l$ for all $j \in V_\epsilon$ **then**
5: $$\lambda^{l+1} \leftarrow \begin{cases} \lambda_j^l + \bar{\lambda_j}, & j \in A_\epsilon \\ \lambda_j^l - \frac{1}{\mu}, & j \in V_\epsilon \\ \lambda_j^l, & j \notin A_\epsilon \ \& \ j \notin V_\epsilon \end{cases}$$
6: $\eta^{l+1} \leftarrow \eta^l (\mu^l)^{0.9}$
7: **else**
8: $\mu_{l+1} \leftarrow \frac{\mu_l}{10}$
9: $\eta^{l+1} \leftarrow \frac{(\mu^{l+1})^{0.1}}{10}$
10: **end if**
11: $l \leftarrow l + 1$
12: Go to 2.

6 Computational results

In this section, we introduce three sets of test problems for the comparison between five variants of NOMAD: each version uses a different approach to solve Subproblem (2). The results of this comparison are detailed in Subsections 6.2, 6.3 and 6.4.

6.1 Test problems

We use three sets of optimization problems of the form (1): the first set is composed of 42 unconstrained problems, the second consists of 19 constrained problems and the third contains 4 simulation-based applications: two MDO problems called AIRCRAFT_RANGE and SIMPLIFIED_WING, a structural engineering design problem of a welded beam called WELDED and a blackbox optimization problem (LOCKWOOD) which minimizes the cost of managing wells in the Lockwood Solvent Groundwater Plume Site (Montana) that prevents the contamination of the Yellowstone River. Table 1 regroups the name, dimension and number of constraints of the 65 test problems.

We implemented the three methods of Sections 3, 4 and 5 into the NOMAD package [34] (www.gerad.ca/nomad) and compare five algorithmic variants. The first variant is denoted MADS and uses the current version of NOMAD (version 3.7) to solve Subproblem (2). The three variants l_1EPF, AUGLAG and l_1AUGLAG respectively use the l_1 exact penalty function, the augmented Lagrangian and the l_1 augmented Lagrangian to treat (2). Finally, the last variant uses the IPOPT package [51] which is software for nonlinear optimization. The tests were conducted on a Linux machine with an Intel(R) Core(TM) i7-2600 (3.40 GHz) processor.

To compare these five methods, we rely on the Moré & Wild performance and data profiles [42] by considering the following formula as the main convergence test:

$$f(x_f) - f(x) \geq (1 - \tau)(f(x_f) - f^*) \tag{27}$$

where τ is the convergence precision, f^* is the best solution found by all solvers for a specific problem and a given budget (of either evaluations or time) and x_f is the first feasible point of the problem found by any solver. If a solver cannot find a feasible solution, the convergence test will always fail which will favor all the solvers that find at least one feasible point. Performance profiles show the ratio of solved problems according to a performance ratio α defined as the upper bound of the number of evaluations needed to satisfy the convergence test. These profiles allow one to compare the relative performances of the algorithms. Data profiles, on the other hand, indicate which algorithms is best for a fixed number of function evaluations: they show the ratio of solved problems to groups of $n + 1$ function evaluations.

Table 1: Description of the test problems.

Unconstrained								Constrained			
Name	n	m	Source	Name	n	m	Source	Name	n	m	Source
ACKLEY	10	0	[29]	POWELLSG	12	0	[24]	CRESCENT10	10	2	[5]
ARWHEAD	10	0	[24]	POWELLSG	20	0	[24]	DISK10	10	1	[5]
ARWHEAD	20	0	[24]	RADAR	7	0	[39]	G1	13	9	[38]
BDQRTIC	10	0	[24]	RANA	2	0	[32]	G2	10	2	[38]
BDQRTIC	20	0	[24]	RASTRIGIN	2	0	[29]	G2	20	2	[38]
BIGGS6	6	0	[24]	RHEOLOGY	3	0	[6]	G4	5	6	[38]
BRANIN	2	0	[29]	ROSENBROCK	2	0	[32]	G6	2	2	[38]
BROWNAL	10	0	[24]	SCHWEFEL	2	0	[47]	G7	10	8	[38]
BROWNAL	20	0	[24]	SHOR	5	0	[36]	G8	2	2	[38]
DIFF2	2	0	[18]	SROSENBR	10	0	[24]	G9	7	4	[38]
ELATTAR	6	0	[36]	SROSENBR	20	0	[24]	G10	8	6	[38]
EVD61	6	0	[36]	TREFETHEN	2	0	[32]	HS44	4	6	[31]
FILTER	9	0	[36]	TRIDIA	10	0	[24]	HS64	3	1	[31]
GRIEWANK	2	0	[29]	TRIDIA	20	0	[24]	HS93	6	2	[31]
GRIEWANK	10	0	[29]	VARDIM	10	0	[24]	HS108	9	13	[31]
HS78	5	0	[36]	VARDIM	20	0	[24]	HS114	9	6	[36]
OSBORNE2	11	0	[36]	WATSON	12	0	[32]	MAD6	5	7	[36]
PBC1	5	0	[36]	WONG1	7	0	[36]	PENTAGON	6	15	[36]
PENALTY1	10	0	[24]	WONG2	10	0	[36]	SNAKE	2	2	[5]
PENALTY1	20	0	[24]	WOODS	12	0	[24]				
POLAK2	10	0	[36]	WOODS	20	0	[24]				
Simulation-based MDO applications											
AIRCRAFT_RANGE	10	10	[1, 48]	SIMPLIFIED_WING	7	3	[50]	WELDED	4	6	[22]
LOCKWOOD	6	4	[33, 37]								

6.2 Unconstrained test set

Problems of the first test set are unconstrained. It implies that penalty parameters and Lagrange multipliers are not necessary and our methods reduce to their inner iterations: the l_1EPF and l_1AUGLAG become an iterative line-search method that uses matrix decompositions to select the descent direction, while the AUGLAG method solves a trust-region subproblem with a combination of projected and conjugate gradient methods.

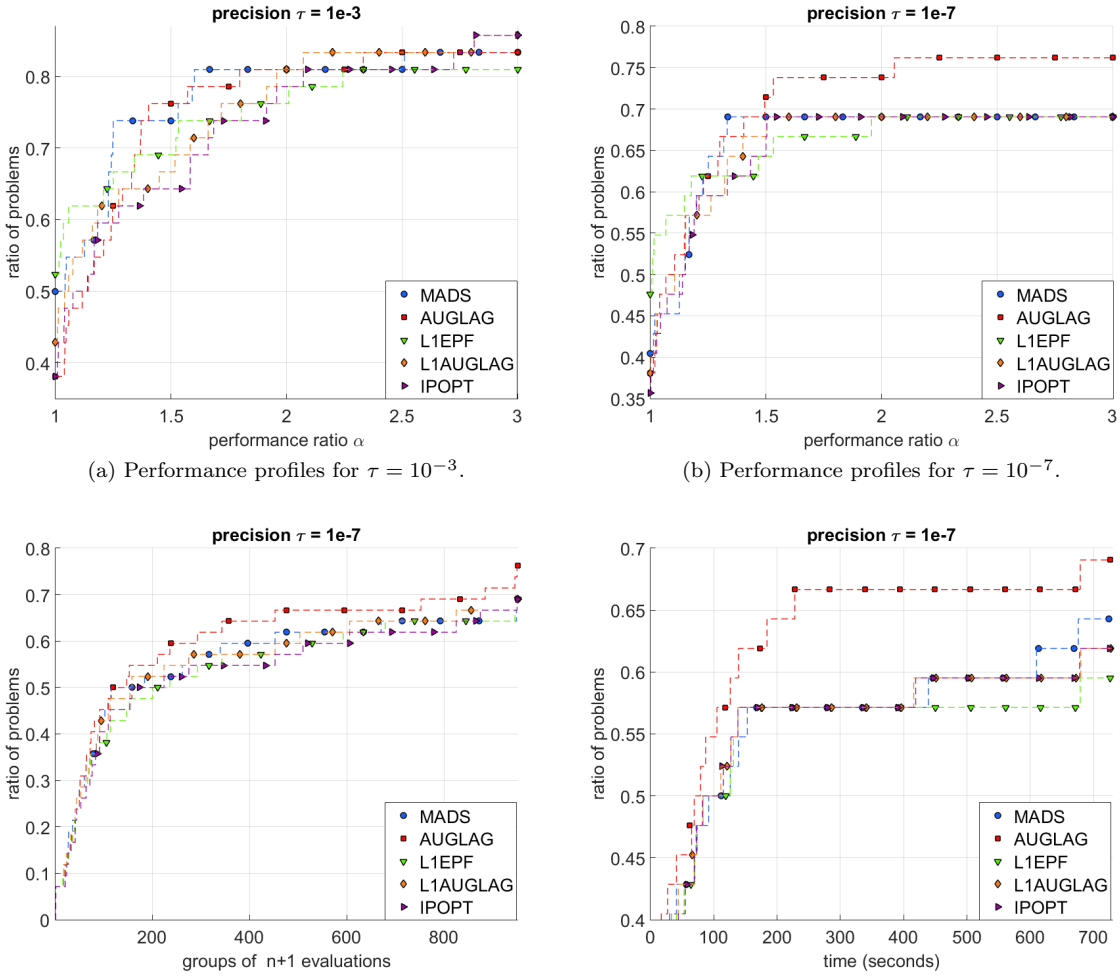
Figure 2(a) regroups performance profiles for the unconstrained set with $\tau = 10^{-3}$. It is difficult to set a winner apart from this graph: for example, IPOPT starts slowly, but emerges on top for a performance ratio $\alpha = 3$. Figures 2(b) and 2(c) represent performance and data profiles for the first set of problems with a precision $\tau = 10^{-7}$. AUGLAG clearly outperforms the other methods in this case.

Figure 2(d) regroups time data profiles: they show the ratio of solved problems with precision $\tau = 10^{-7}$ for a given time budget. This figure takes into account the effort deployed to solve the quadratic subproblems and confirms the superiority of AUGLAG for unconstrained optimization. We conclude that a trust-region is preferable to a line-search method to solve the augmented Lagrangian subproblem in the absence of constraints in Problem (2).

Even if the l_1EPF and l_1AUGLAG reduce to the same inner iteration, we can see that there are differences in the profiles because these algorithms choose different starting points for the subproblem. This implementation choice was done after several preliminary tests on constrained problems and was kept for the unconstrained case. Still, the variations between l_1EPF and l_1AUGLAG are small and the two methods seem equivalent.

6.3 Constrained test set

The problems of the second set are constrained. Figure 3 shows performance and data profiles for precisions $\tau = 10^{-3}$ and $\tau = 10^{-7}$. For this set of problems, all the new methods outperform MADS, especially l_1AUGLAG which gives the best results and solves almost 95% of the problems while the MADS-based NOMAD solves only 75% when $\tau = 10^{-7}$ as illustrated in the performance profiles of Figure 3(a).



(a) Performance profiles for $\tau = 10^{-3}$.

(b) Performance profiles for $\tau = 10^{-7}$.

(c) Data profiles for $\tau = 10^{-7}$.

(d) Time data profiles for $\tau = 10^{-7}$.

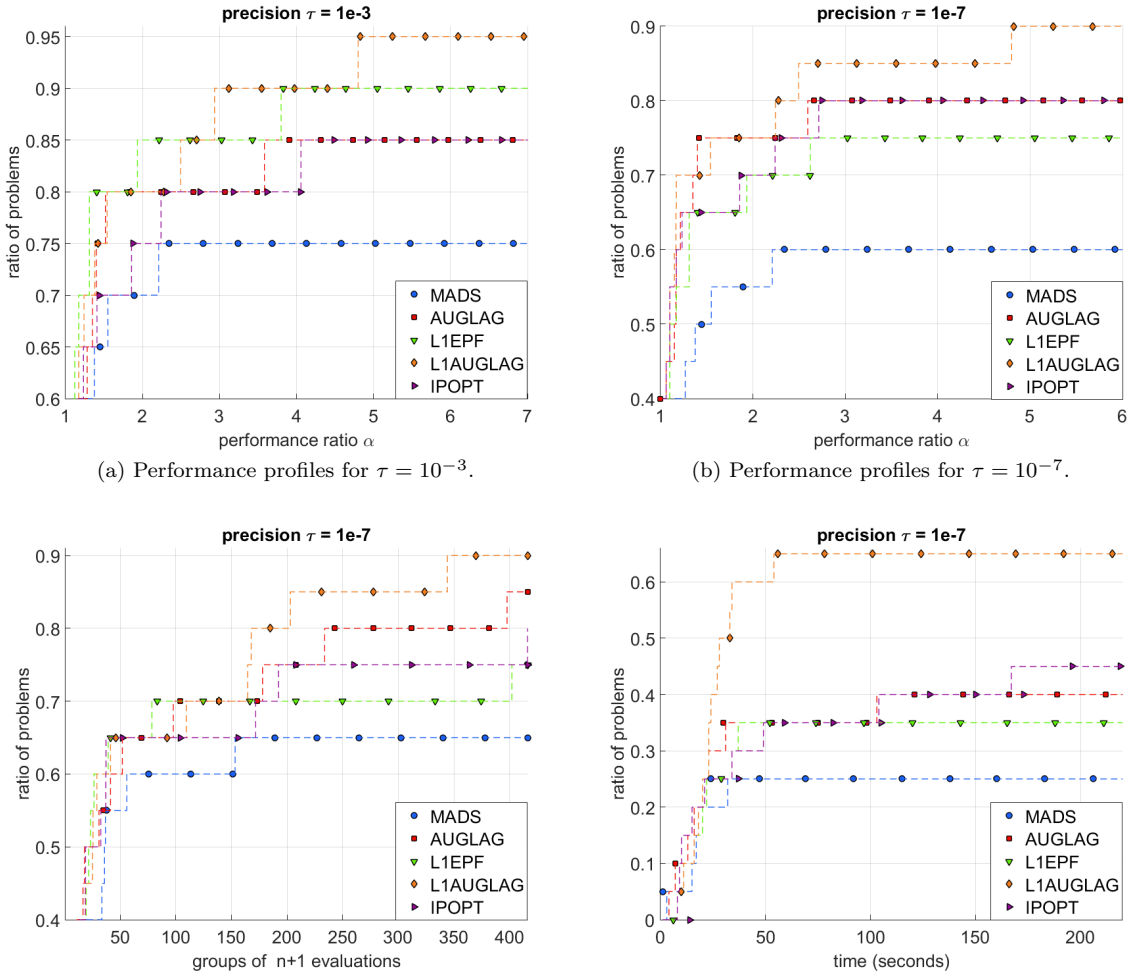
Figure 2: Performance and data profiles on the unconstrained test set.

Figure 3(d) represents time data profiles for the second set of test problems: using IPOPT to solve subproblems seems to be efficient timewise, but still, the figure confirms that l_1AUGLAG is the best approach for the constrained case.

6.4 Simulation-based applications

For the last set, we create 50 different instances for each of the four applications: for a given problem, a Latin hypercube sampling generates 50 starting points and each of these instances is solved by the five variants of NOMAD. Some of these starting points are feasible, others are not. These simulated-based problems tend to be timewise computationally expensive: the 750 tests of AIRCRAFT_RANGE, SIMPLIFIED_WING and WELDED ran for over five days and the 250 runs of LOCKWOOD, took almost two weeks of execution time.

For each application, we draw data and performance profiles to compare the different methods (we consider that all the instances are different problems). It is important to carefully select an appropriate value of the precision parameter τ for each problem since, on one hand, if τ is too small, the profiles tends to pile up with a small proportion of problems solved. On the other hand, if τ is too large, the curves converge to 100% problem solved quickly. For clarity, we chose values of τ as the power of 10 that spreads out as much as possible the profiles.



(a) Performance profiles for $\tau = 10^{-3}$.

(b) Performance profiles for $\tau = 10^{-7}$.

(c) Data profiles for $\tau = 10^{-7}$.

(d) Time data profiles for $\tau = 10^{-7}$.

Figure 3: Performance and data profiles on the constrained test set.

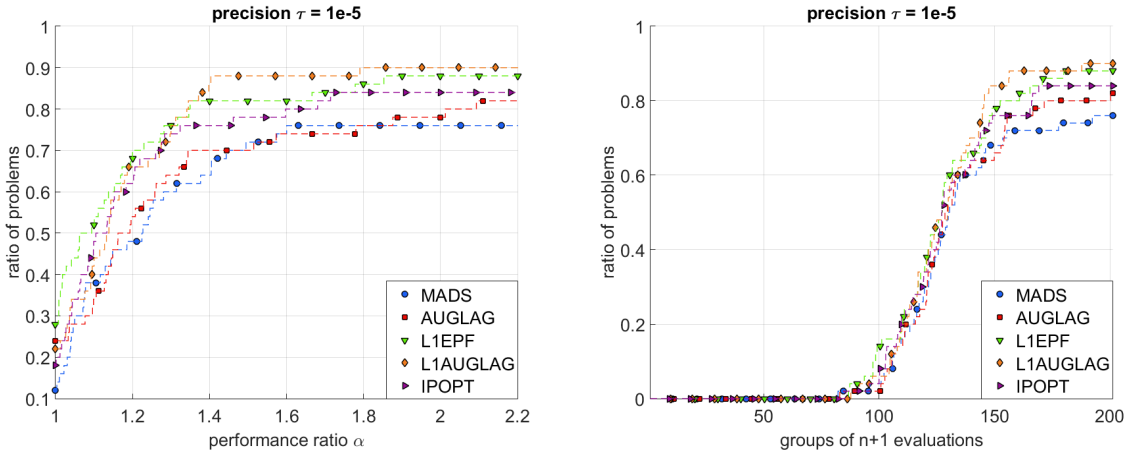
In Figures 4 and 5, we can immediately see that l_1AUGLAG is the best approach for the AIRCRAFT_RANGE and LOCKWOOD problems. But, while l_1EPF and IPOPT seem to give good results on the AIRCRAFT_RANGE application (they are even overlapping l_1AUGLAG for $\alpha \leq 1.3$), AUGLAG and MADS are more efficient on the LOCKWOOD problem.

In Figure 6, MADS shows some good results when $\alpha \leq 4.5$ and the budget is limited to 300 $(n+1)$ evaluations. When we allow more evaluation budget, l_1AUGLAG can almost reach 80% of problems solved.

In Figure 7, the AUGLAG approach outperforms the rest of the algorithms with IPOPT coming last again.

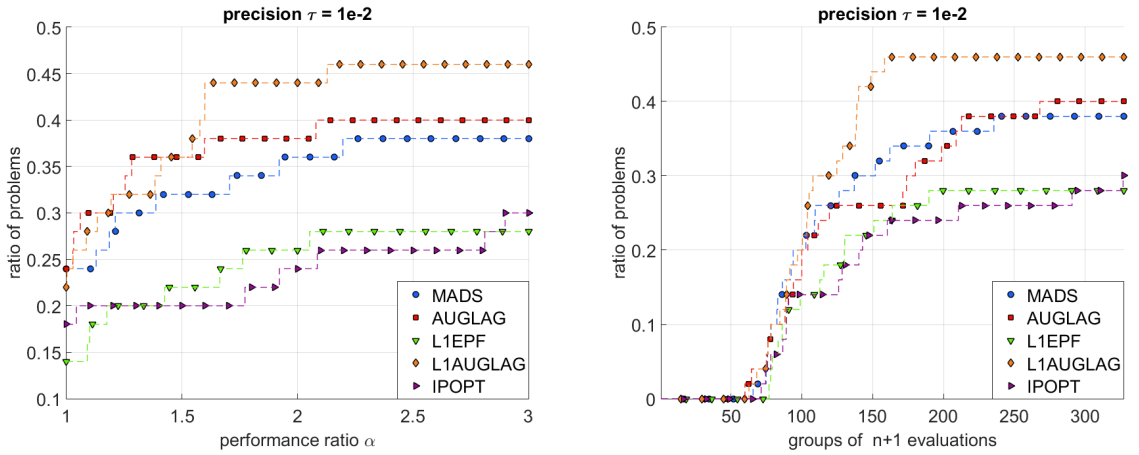
These profiles suggest that IPOPT should be discarded as a replacement for MADS in solving the quadratic subproblem. To recommend an alternative for MADS among the three proposed methods, we build performance and data profiles with $\tau = 10^{-2}$ and $\tau = 10^{-5}$ for the combined 200 instances of the four simulation-based applications.

In Figure 8, l_1AUGLAG seems to be the clear winner. As suspected, IPOPT is not a viable option to solve the subproblem, while the three remaining approaches, MADS, l_1EPF and AUGLAG, are competitive with each other on the simulation-based problems. This was predictable since IPOPT is used for general nonlinear optimization while methods such as the l_1AUGLAG have an advantage when solving quadratically constrained quadratic problems.



(a) Performance profiles. (b) Data profiles.

Figure 4: Performance and data profiles of the AIRCRAFT_RANGE problem.



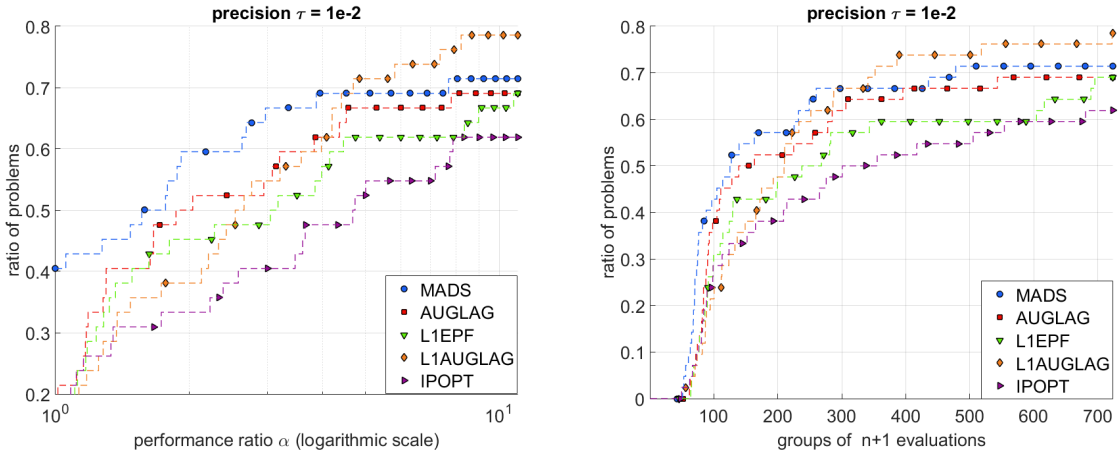
(a) Performance profiles. (b) Data profiles.

Figure 5: Performance and data profiles of the LOCKWOOD problem.

7 Discussion

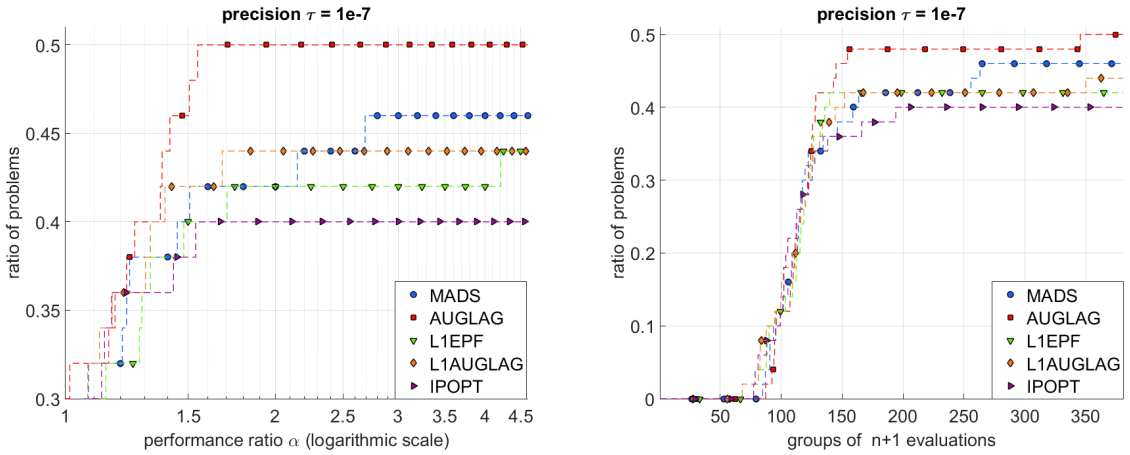
This work details the implementation of two existing nonlinear optimization methods (l_1EPF and AUGLAG) and the development of a new algorithm called the l_1 augmented Lagrangian method (l_1AUGLAG). These methods are used within a derivative-free optimization framework to solve quadratically constrained quadratic subproblems that arise when using quadratic models of the objective function and of the constraints. It would be possible to use the implemented methods within algorithms that rely on treating subproblems to solve general nonlinear problems with derivatives. Another avenue would be to use the augmented Lagrangian method to transform a DFO problem with general equality constraints into an unconstrained problem that MADS can deal with.

This paper shows that using a dedicated algorithm to solve quadratic subproblems improves the overall performance of MADS with quadratic models. The results show that, in the unconstrained case, using a trust-region approach (inner iteration of the augmented Lagrangian) yields better result than line-search method (which is the case for both the l_1 exact penalty function and the l_1 augmented Lagrangian).



(a) Performance profiles. (b) Data profiles.

Figure 6: Performance and data profiles of the SIMPLIFIED_WING problem.



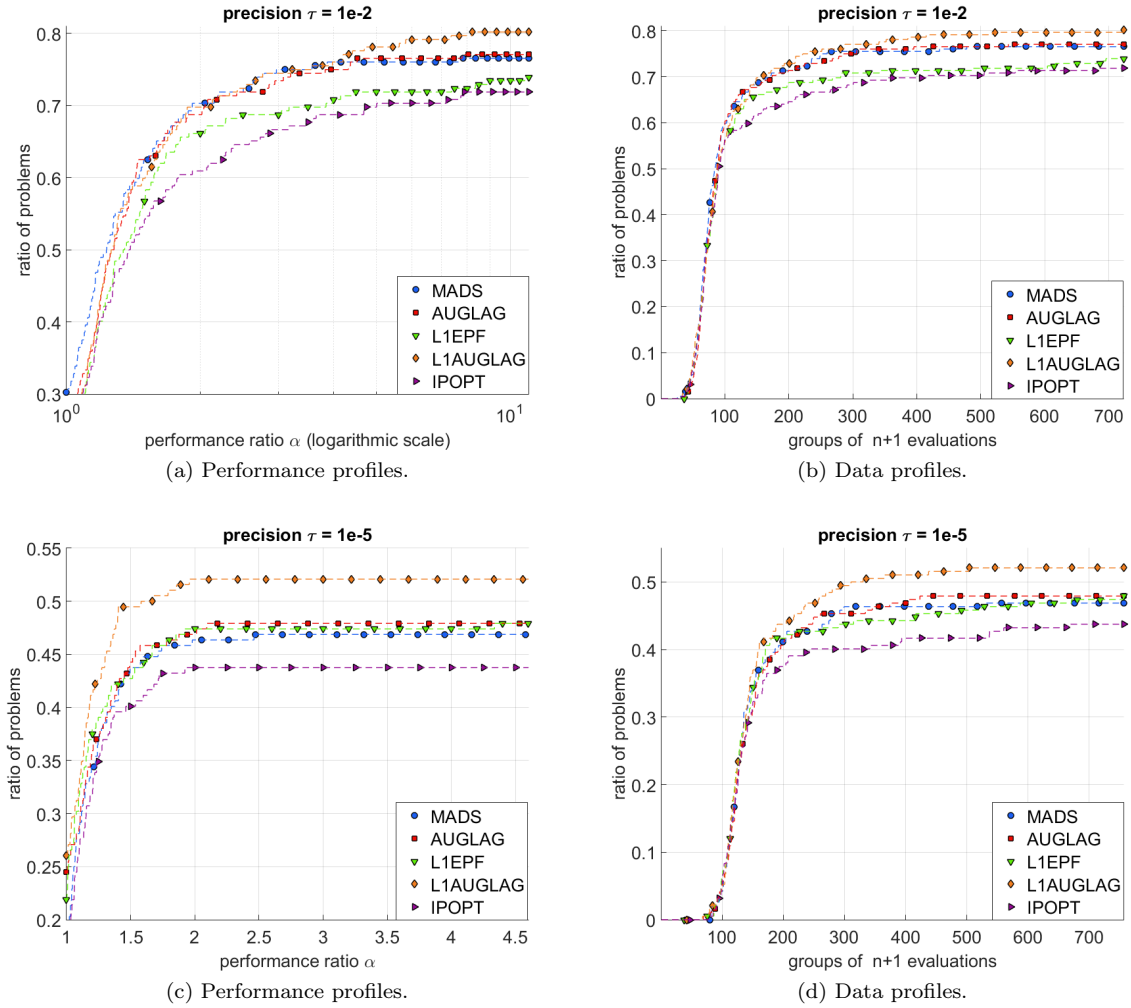
(a) Performance profiles. (b) Data profiles.

Figure 7: Performance and data profiles of the WELDED problem.

The l_1AUGLAG approach gives the best results in the constrained case as, we believe, it combines the strengths of both AUGLAG and l_1EPF methods: the Lagrange multipliers terms improve the computation performances and the l_1 penalty term allows Problem (16) to have a piecewise quadratic structure.

In all cases, there is at least one method that improves over MADS in regards of both results and time and this is the main goal of the paper. The computational results show also that at least one of the three suggested methods is performing better than IPOPT which validates the choice of implementing a dedicated method for NOMAD.

We conclude with recommendations for the NOMAD software package. For solving quadratic subproblems in the search step, NOMAD should use the Moré & Toraldo augmented Lagrangian algorithm in the unconstrained case and the l_1 augmented Lagrangian method for constrained problems.



(a) Performance profiles.

(b) Data profiles.

(c) Performance profiles.

(d) Data profiles.

Figure 8: Performance and data profiles for all 200 instances of the simulation-based applications.

References

[1] J.S. Agte, J. Sobieszczanski-Sobieski, and R.R.J. Sandusky. Supersonic business jet design through bilevel integrated system synthesis. In Proceedings of the World Aviation Conference, volume SAE Paper No. 1999-01-5622, San Francisco, CA, 1999. MCB University Press, Bradford, UK.

[2] S. Arreckx, A. Lambe, J.R.R.A. Martins, and D. Orban. A matrix-free augmented lagrangian algorithm with application to large-scale structural design optimization. Optimization and Engineering, 17(2):359–384, 2016.

[3] C. Audet, V. Béchard, and S. Le Digabel. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. Journal of Global Optimization, 41(2):299–318, 2008.

[4] C. Audet and J.E. Dennis, Jr. Mesh adaptive direct search algorithms for constrained optimization. SIAM Journal on Optimization, 17(1):188–217, 2006.

[5] C. Audet and J.E. Dennis, Jr. A Progressive Barrier for Derivative-Free Nonlinear Programming. SIAM Journal on Optimization, 20(1):445–472, 2009.

[6] C. Audet and W. Hare. Derivative-Free and Blackbox Optimization, In preparation.

[7] C. Audet, M. Kokkolaras, S. Le Digabel, and B. Talgorn. Order-based error for managing ensembles of surrogates in derivative-free optimization. Technical Report G-2016-36, Les cahiers du GERAD, 2016.

[8] C. Audet, S. Le Digabel, and M. Peyrega. A derivative-free trust-region augmented Lagrangian algorithm. Technical Report G-2016-53, Les cahiers du GERAD, 2016.

[9] A.J. Booker, J.E. Dennis, Jr., P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset. A Rigorous Framework for Optimization of Expensive Functions by Surrogates. Structural and Multidisciplinary Optimization, 17(1):1–13, 1999.

[10] C. Charalambous. A lower bound for the controlling parameters of the exact penalty functions. Mathematical programming, 15(1):278–290, 1978.

[11] T.F. Coleman and A.R. Conn. Second-order conditions for an exact penalty function. Mathematical Programming, 19(1):178–185, 1980.

[12] T.F. Coleman and A.R. Conn. Nonlinear programming via an exact penalty function: Asymptotic analysis. Mathematical programming, 24(1):123–136, 1982.

[13] T.F. Coleman and A.R. Conn. Nonlinear programming via an exact penalty function: Global analysis. Mathematical Programming, 24(1):137–161, 1982.

[14] A.R. Conn, N.I.M. Gould, and Ph.L. Toint. LANCELOT: a Fortran package for large-scale nonlinear optimization (release A). Springer, New-York, 1992.

[15] A.R. Conn, N.I.M. Gould, and Ph.L. Toint. A note on exploiting structure when using slack variables. Mathematical Programming, 67(1):89–97, 1994.

[16] A.R. Conn, N.I.M. Gould, and Ph.L. Toint. Numerical experiments with the LANCELOT package (release A) for large-scale nonlinear optimization. Mathematical Programming, 73(1):73–110, 1996.

[17] A.R. Conn, N.I.M. Gould, and Ph.L. Toint. Trust region methods. SIAM, 2000.

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

[19] A.R. Conn, K. Scheinberg, and L.N. Vicente. Introduction to Derivative-Free Optimization. MOS-SIAM Series on Optimization. SIAM, Philadelphia, 2009.

[20] A.R. Conn, L.N. Vicente, and C. Visweswariah. Two-step algorithms for nonlinear optimization with structured applications. SIAM Journal on Optimization, 9(4):924–947, 1999.

[21] C. Fortin and H. Wolkowicz. The trust region subproblem and semidefinite programming. Optimization Methods and Software, 19(1):41–67, 2004.

[22] H. Garg. Solving structural engineering design optimization problems using an artificial bee colony algorithm. Journal of Industrial and Management Optimization, 10(3):777–794, 2014.

[23] N.I.M. Gould, S. Lucidi, and Ph.L. Toint. Solving the trust-region subproblem using the Lanczos method. SIAM Journal on Optimization, 9(2):504–525, 1999.

[24] N.I.M. Gould, D. Orban, and Ph.L. Toint. CUTEr (and SifDec): A constrained and unconstrained testing environment, revisited. ACM Transactions on Mathematical Software, 29(4):373–394, 2003.

[25] N.I.M. Gould, D.P. Robinson, and H.S. Thorne. On solving trust-region and other regularised subproblems in optimization. Mathematical Programming Computation, 2(1):21–57, 2010.

[26] R.B. Gramacy, G.A. Gray, S. Le Digabel, H.K.H. Lee, P. Ranjan, G. Wells, and S.M. Wild. Modeling an Augmented Lagrangian for Blackbox Constrained Optimization. Technometrics, 58(1):1–11, 2016.

[27] I. Griva, S.G. Nash, and A. Sofer. Linear and Nonlinear Optimization. Society for Industrial and Applied Mathematics, 2009.

[28] W.W. Hager. Minimizing a quadratic over a sphere. SIAM Journal on Optimization, 12(1):188–208, 2001.

[29] A.-R. Hedar. Global optimization test problems. http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm.

[30] M.R. Hestenes. Multiplier and gradient methods. Journal of Optimization Theory and Applications, 4(5):303–320, 1969.

[31] W. Hock and K. Schittkowski. Test Examples for Nonlinear Programming Codes, volume 187 of Lecture Notes in Economics and Mathematical Systems. Springer, Berlin, Germany, 1981.

[32] M. Jamil and X.-S. Yang. A literature survey of benchmark functions for global optimisation problems. International Journal of Mathematical Modelling and Numerical Optimisation, 4(2):150–194, 2013.

[33] A. Kannan and S.M. Wild. Benefits of Deeper Analysis in Simulation-based Groundwater Optimization Problems. In Proceedings of the XIX International Conference on Computational Methods in Water Resources (CMWR 2012), June 2012.

[34] S. Le Digabel. Algorithm 909: NOMAD: Nonlinear Optimization with the MADS algorithm. ACM Transactions on Mathematical Software, 37(4):44:1–44:15, 2011.

[35] D.G. Luenberger. Control problems with kinks. IEEE Transactions on Automatic Control, 15(5):570–575, 1970.

[36] L. Lukšan and J. Vlček. Test problems for nonsmooth unconstrained and linearly constrained optimization. Technical Report V-798, ICS AS CR, 2000.

[37] L.S. Matott, A.J. Rabideau, and J.R. Craig. Pump-and-treat optimization using analytic element method flow models. Advances in Water Resources, 29(5):760–775, 2006.

[38] Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. Evolutionary computation, 4(1):1–32, 1996.

[39] N. Mladenović, J. Petrović, V. Kovačević-Vujčić, and M. Čangalović. Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. European Journal of Operational Research, 151(2):389–399, 2003.

[40] J.J. Moré and D.C. Sorensen. Computing a trust region step. SIAM Journal on Scientific Computing, 4(3):553–572, 1983.

[41] J.J. Moré and G. Toraldo. On the solution of large quadratic programming problems with bound constraints. SIAM Journal on Optimization, 1(1):93–113, 1991.

[42] J.J. Moré and S.M. Wild. Benchmarking derivative-free optimization algorithms. SIAM Journal on Optimization, 20(1):172–191, 2009.

[43] J. Nocedal and S.J. Wright. Numerical Optimization. Springer Series in Operations Research. Springer, New York, 1999.

[44] T.K. Pong and H. Wolkowicz. The generalized trust region subproblem. Computational Optimization and Applications, 58(2):273–322, 2014.

[45] M.J.D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, Optimization, pages 283–298. Academic Press, New York, 1969.

[46] F. Rendl and H. Wolkowicz. A semidefinite framework for trust region subproblems with applications to large scale minimization. Mathematical Programming, 77(1):273–299, 1997.

[47] H.-P. Schwefel. Numerical Optimization of Computer Models. John Wiley & Sons, Inc., New York, NY, USA, 1981.

[48] J. Sobieszczanski-Sobieski, J.S. Agte, and R.R. Sandusky, Jr. Bi-Level Integrated System Synthesis (BLISS). Technical Report NASA/TM-1998-208715, NASA, Langley Research Center, 1998.

[49] T. Pietrzykowski. An exact potential method for constrained maxima. SIAM Journal on numerical analysis, 6(2):299–304, 1969.

[50] C. Tribes, J.-F. Dubé, and J.-Y. Trépanier. Decomposition of multidisciplinary optimization problems: formulations and application to a simplified wing design. Engineering Optimization, 37(8):775–796, 2005.

[51] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Mathematical Programming, 106(1):25–57, 2006.

[52] W.I. Zangwill. Non-linear programming via penalty functions. Management science, 13(5):344–358, 1967.