A trust-region derivative-free algorithm for constrained optimization

P.D. Conejo∗ and E.W. Karas† and L.G. Pedroso‡

February 26, 2014

Abstract

We propose a trust-region algorithm for constrained optimization problems in which the derivatives of the objective function are not available. In each iteration, the objective function is approximated by a model obtained by quadratic interpolation, which is then minimized within the intersection of the feasible set with the trust region. Since the constraints are handled in the trust-region subproblems, all the iterates are feasible even if some interpolation points are not. The rules for constructing and updating the quadratic model and the interpolation set use ideas from the BOBYQA software, a well-succeeded algorithm for box-constrained problems. The subproblems are solved by ALGENCAN, a competitive implementation of an Augmented Lagrangian approach for general constrained problems. Some numerical results for the Hock-Schittkowski collection are presented, followed by a performance comparison among our proposal and three derivative-free algorithms found in the literature.

1 Introduction

We propose a trust-region algorithm for solving the problem

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in \Omega
\end{align*}
\]

where \(f : \mathbb{R}^n \rightarrow \mathbb{R}\) and \(\Omega = \{x \in \mathbb{R}^n \mid h(x) = 0, g(x) \leq 0\}\), with \(h : \mathbb{R}^n \rightarrow \mathbb{R}^p\) and \(g : \mathbb{R}^n \rightarrow \mathbb{R}^q\) differentiable functions. Although our algorithm can be applied when the objective function is nonsmooth, it was designed for the class of problems in which \(f\) is smooth but its derivatives are not available.

Derivative-free methods for continuous optimization have received increasing attention from the scientific community due to the wide range of problems where derivatives cannot be explicitly computed [8, 10], as for example when the objective function is provided by a simulation package or a black box. Considering the class of trust-region derivative-free methods, many algorithms can be found in the literature. For the unconstrained case, i.e., when \(\Omega = \mathbb{R}^n\), Conn, Scheinberg and Vicente [10], Conn, Scheinberg and Toint [7], Scheinberg and Toint [26] and Powell [25] presented algorithms with global convergence results, while Fasano, Morales and Nocedal [13], Conn and Toint [11], and Powell [21, 22] discussed algorithms with good practical performance. Conejo et al. proposed a globally convergent algorithm for convex constrained problems. Powell has contributed largely in the area of derivative-free methods. For problems with box constraints, Powell proposed BOBYQA software [23] and discussed

∗State University of Paraná, Dep. of Mathematics, Cascavel, PR, Brazil.
†Federal University of Paraná, Dep. of Mathematics, Curitiba, PR, Brazil. This author was partially supported by CNPq, Brazil.
‡Federal University of Paraná, Dep. of Mathematics, Curitiba, PR, Brazil.
an algorithm for linear constrained problems in [24]. Aroux, Echebest and Pilotta [3] and Gratton, Toint and Tröltzsch [15] proposed active-set algorithms for bound-constrained optimization. For general constrained problems, which is the focus of the present paper, it is important to mention the algorithm introduced by Conn, Scheinberg and Toint [8]. The implementation of this algorithm, called DFO, is competitive even when compared to very recent approaches.

The classical trust-region framework [6] for unconstrained problems consists in, at each iteration, approximate the objective function by a quadratic model and minimize it within the trust-region, which is a ball centered at the current iterate with suitable radius. In derivative-free techniques, the models are usually constructed by quadratic interpolation [5, 9, 10]. Our approach uses the structure presented in the BOBYQA software [23] for constructing and updating the model and the interpolation set. As in the BOBYQA algorithm, there is freedom in the quantity of interpolation points, which is chosen by the user and fixed through the iterations. This freedom would result in the non-unicity of the interpolation model. However, among all the models that interpolate the objective function in the current interpolation set, we choose the one that has Hessian matrix as close as possible in the Frobenius norm from the Hessian matrix of the previous model, and with this choice the model is uniquely defined. The subproblem, that consists in minimizing the quadratic model within the intersection of the feasible set and the trust region, is solved by ALGENCAN, which is an implementation of the Augmented Lagrangian algorithm proposed in [1, 2]. Thus the constraints are treated in the subproblems and the algorithm generates a sequence of feasible iterates. A new iterate is incorporated in the interpolation set whenever it provides a decrease in the objective function.

The paper is organized as follows. In Section 2, we present our trust-region derivative-free algorithm. Section 3 discusses the construction and update of the interpolation set and model. Numerical results are presented in Section 4. Finally, we state some conclusions.

Notation

$\mathcal{P}^2_n$ denotes the set of all quadratic polynomials in $\mathbb{R}^n$.

$e^i$ denotes the $i$-th coordinate vector.

$\| \cdot \|$ denotes the Euclidean norm.

$Q(Y) = f(Y)$ means that $Q(y^i) = f(y^i)$ for all $y^i \in Y$.

2 The algorithm

In this section we state our algorithm, postponing to the next section the discussion about the details of its steps.

At each iteration $k$, we construct a quadratic model $Q_k$ by polynomial interpolation. Given $x^b \in \mathbb{R}^n$, the quadratic model assumes the form

$$Q_k(x) = c_k + g_k^T (x - x^b) + \frac{1}{2} (x - x^b)^T G_k (x - x^b),$$

(2)

where $c_k \in \mathbb{R}$, $g_k \in \mathbb{R}^n$ and $G_k \in \mathbb{R}^{n \times n}$ is a symmetric matrix.

The number of interpolation points,

$$m \in [2n + 1, \overline{m}] \quad \text{with} \quad \overline{m} = \frac{1}{2} (n + 1)(n + 2),$$

is chosen by the user and fixed through the iterations. We denote the current interpolation set by $Y_k = \{y^1, y^2, \ldots, y^m\} \subset \mathbb{R}^n$.

As usual in trust-region frameworks, our algorithm considers a sequence of subproblems consisting in minimizing the model $Q_k$ within the trust region with radius $\Delta_k > 0$ centered at the current point $x_k$. 

2
Furthermore, the constraints of problem (1) are incorporated in the subproblem, that takes the form

\[
\begin{align*}
\text{minimize} & \quad Q_k(x) \\
\text{subject to} & \quad x \in \Omega \\
& \quad \|x - x_k\|_\infty \leq \Delta_k.
\end{align*}
\]

The algorithm proposed in this paper is defined as follows.

**Algorithm 1**

Data: \(x_1 \in \Omega, \epsilon > 0, \rho_1 > \epsilon, \gamma \in (0, 1), m \in [2n+1, m], s > 2, \Delta_1 = \rho_1\).

**Step 1:** Constructing the model
Set \(y_1 = x_k \) and \(x^0 = x_k\).
Construct the interpolation set \(Y_k\) with radius \(\rho = \rho_k\).
Construct the model \(Q_k\) by polynomial interpolation.

**Step 2:** Solving the subproblem
Compute \(x^*\), solution of the subproblem (3).

**Step 3:** Updates
If \(\|x^* - x_k\| \leq 0.5\rho_k\), then
If \(\rho_k \leq \epsilon\), then Stop.
If \(\max_{1 \leq j \leq m} \|y_j - x_k\| > s\rho_k\), then \(\rho_{k+1} = \rho_k\).
Else, \(\rho_{k+1} = \gamma\rho_k\).
\(x_{k+1} = x_k, \Delta_{k+1} = \Delta_k, k = k + 1\) and go to Step 1.
Else,
\[
r = \frac{f(x_k) - f(x^*)}{Q_k(x_k) - Q_k(x^*)},
\]
\[
\Delta = \begin{cases} 
0.5\Delta_k, & \text{if } r < 0.1, \\
\Delta_k, & \text{if } 0.1 \leq r \leq 0.7, \\
2\Delta_k, & \text{if } r > 0.7.
\end{cases}
\]
If \(f(x^*) < f(x_k)\), then
Choose \(y^*\) that will leave the interpolation set \(Y_k\).
Set \(Y^* = Y_k \setminus \{y^*\} \cup \{x^*\}\).
If \(r \geq 0.1\), then
\(x_{k+1} = x^*, \Delta_{k+1} = \Delta, \rho_{k+1} = \rho_k, Y_{k+1} = Y^*\).
Compute \(Q_{k+1}\) by solving (5), \(k = k + 1\) and go to Step 2.
Else,
If \(Y^*\) is not sufficiently posed or \(\max_{1 \leq j \leq m} \|y_j - x_k\| > s\rho_k\), then
\(x_{k+1} = x_k, \Delta_{k+1} = \Delta, \rho_{k+1} = \rho_k, k = k + 1\) and go to Step 1.
Else,
If \(\rho_k \leq \epsilon\), then Stop.
Else,
If \(f(x^*) < f(x_k)\), then
\(Y_{k+1} = Y^*\) and compute \(Q_{k+1}\) by solving (5).
Else,
\(x_{k+1} = x_k, Q_{k+1} = Q_k\).
\(x_{k+1} = x_k, \Delta_{k+1} = \Delta, \rho_{k+1} = \rho_k, \gamma\rho_k, k = k + 1\) and go to Step 2.
Note that the sequence \((x_k)\) generated by the algorithm is feasible, but the interpolation points can be infeasible. The point \(x^*\), obtained as a solution of the subproblem (3), will be incorporated to the interpolation set whenever this point provides a simple decrease in the objective function, unless the resulting interpolation set is not sufficiently posed. The meaning of the expression sufficiently posed will be explained in the next section. If the decrease at \(x^*\) is sufficient, in the sense that \(r \geq 0.1\), then this point is accepted as the new iterate. The scalar \(r\) is the ratio between the actual reduction and the one predicted by the model, which is a classical measure in trust-region algorithms. The magnitude of this scalar is used also to decide if the trust-region radius \(\Delta_k\) will be reduced, maintained or increased. The parameter \(\rho_k \leq \Delta_k\) controls the diameter of the interpolation set and is related to the stopping criterion. When \(\|x^* - x_k\| < 0.5\rho_k\) or \(r < 0.1\), the progress in the current iteration is considered insufficient. Then the algorithm might test if it is time to stop, since the iterate may be close to a solution. If this is not the case, at least one of the following measures will be taken: reduce the parameter \(\rho_k\), so the next interpolation sets will tend to have smaller diameters, resulting on more accurate models, or redefine the interpolation set and reconstruct the model at Step 1, since the lack of progress may be due to accumulation of errors from the previous model updates. In order to avoid unnecessary reductions of \(\rho_k\), we perform the test

\[
\max_{1 \leq j \leq m} \left(\|y^j - x_k\|\right) > s\rho_k.
\]

If this condition holds, at least one of the interpolation points is far from the current iterate, then the value of \(\rho_k\) is kept for the next iteration and the interpolation set is reconstructed at Step 1.

3 The model and the interpolation set

In this section we discuss how to construct and update the interpolation set and the model. We would like to emphasize that the procedures discussed in this section are based on [23]. Consider the current interpolation set \(Y_k = \{y^1, y^2, \ldots, y^m\}\). If \(m = \overline{m}\) and \(Y_k\) is posed [10], the interpolation conditions

\[
Q_k(Y_k) = f(Y_k)
\]

determine uniquely the model \(Q_k\) defined in (2). On the other hand, when \(m < \overline{m}\), there might be infinitely many models that verify (4).

In some iterations the model and the interpolation set are constructed from scratch, while in others they are updated from the previous ones. The construction takes place in the first iteration and whenever a lack of progress is detected in the algorithm. In these cases, the interpolation points are chosen differing from the current point in at most two components, so a quadratic model can be easily computed.

During an updating procedure, only one interpolation point of \(Y_k\), say \(y^t\), is modified and replaced by \(x^*\), so \(Y_{k+1} = Y_k \setminus \{y^t\} \cup \{x^*\}\). When computing \(Q_{k+1}\) from \(Q_k\), among all the models that satisfy the interpolation conditions, we choose the one that is as close as possible in the Frobenius norm to the current model \(Q_k\), which means that \(Q_{k+1}\) is the solution of

\[
\begin{align*}
\text{minimize} & \quad \|\nabla^2 Q - G_k\|_F \\
\text{subject to} & \quad Q(Y_{k+1}) = f(Y_{k+1}).
\end{align*}
\]

3.1 Construction

The interpolation set \(Y_k = \{y^1, y^2, \ldots, y^m\}\) is constructed according to rules similar to the ones found in [23]. Given \(y^1 \in \Omega\) and \(\rho > 0\), the next 2\(\nu\) interpolation points are given by

\[
y^{t+1} = y^1 + \rho e^t \quad \text{and} \quad y^{t+i+1} = y^1 - \rho e^i,
\]

for \(1 \leq t \leq \nu - 1\) and \(1 \leq i \leq \nu - 1\).
for \( i = 1, \ldots, n \). The remaining \( m - 2n - 1 \) points are given by
\[
y^j = y^i + pe^{\rho_j},
\]
for \( 2n + 2 \leq j \leq m \), where
\[
u_j = \begin{cases} j - 2n - 1, & 2n + 2 \leq j \leq 3n + 1, \\ u_{j-n}, & 3n + 2 \leq j \leq m \end{cases}
\]
and
\[
v_j = \begin{cases} u_j + c_j & \text{if } (u_j + c_j) \in \{1, \ldots, n\} \\ u_j + c_j - n & \text{if } (u_j + c_j) \notin \{1, \ldots, n\} \end{cases}
\]
with \( c_j = \left\lfloor \frac{j - 2n - 1}{n} \right\rfloor \), where \([a]\) denotes the smallest integer greater than or equal to \( a \). It is worth noticing that this interpolation set is quite simple. It consists of a point \( y^1 \), \( 2n \) points in the coordinate directions from \( y^1 \) and \( m - 2n - 1 \) points differing from \( y^1 \) in two components.

Given the interpolation set \( Y_k = \{y^1, y^2, \ldots, y^m\} \) constructed according to (6)-(9), consider \( x^b = y^1 \). As discussed in [23], the quadratic model that satisfies the interpolation condition \( Q_k(Y_k) = f(Y_k) \) is given by (2) with
\[
c_k = f(y^1),
\]
\[
[g_k]_j = \frac{1}{2\rho^2} \left( f(y^{j+1}) - f(y^{j+n}) \right), \quad j = 1, 2, \ldots, n,
\]
\[
[G_k]_{ij} = \frac{1}{\rho^2} \left( f(y^{j+1}) + f(y^{j+n}) - 2f(y^1) \right), \quad j = 1, 2, \ldots, n,
\]
\[
[G_k]_{u,v} = \frac{1}{\rho^2} \right( f(y^j) - f(y^{j+n}) + f(y^{j+1}) \right), \quad j = 2n + 2, \ldots, m,
\]
\[
[g_k]_{u,v} = 0, \quad j = m + 1, \ldots, m,
\]
\[
[g_k]_{v,u} = \left( G_k \right)_{u,v}, \quad j = 2n + 2, \ldots, m.
\]

The model is constructed with little computational effort due to the simplicity of the constructed interpolation set. Note that the matrix \( G_k \) is diagonal if \( m = 2n + 1 \).

### 3.2 Auxiliary parameters

In this section we define some parameters necessary for the update procedures. The first one is an auxiliary matrix \( H \in \mathbb{R}^{(m+n+1) \times (m+n+1)} \). During a construction procedure, we define the matrix \( H \) as
\[
H = \begin{pmatrix} ZZ^T & E^T \\ E & 0 \end{pmatrix}
\]
with \( Z \in \mathbb{R}^{m \times (m-n-1)} \) and \( E \in \mathbb{R}^{(n+1) \times m} \) given by
\[
Z_{1,j} = -\frac{\sqrt{\frac{n}{\rho^2}}}{\sqrt{\frac{n}{\rho^2}}}, \quad Z_{j+1,j} = Z_{m+j+1,j} = \frac{\sqrt{\frac{n}{\rho^2}}}{\sqrt{\frac{n}{\rho^2}}}, \quad j = 1, \ldots, n,
\]
\[
Z_{1,j-n-1} = Z_{j,j-n-1} = \frac{1}{\rho^2}, \quad j = 2n + 2, \ldots, m,
\]
\[
Z_{u+1,j-n-1} = Z_{u+1,j-n-1} = -\frac{1}{\rho^2}, \quad j = 2n + 2, \ldots, m,
\]
\[
E_{1,1} = 1, \quad E_{i,j} = \frac{1}{2\rho}, \quad E_{k,n} = -\frac{1}{2\rho}, \quad j = 2, \ldots, n + 1,
\]
the remaining elements of \( E \) are null.
Note that this matrix was not used in the previous section to construct neither the model nor the interpolation set, but it has to be defined since it will be necessary ahead. At an update, once defined $t$ such that $y^t$ will leave the interpolation set, the new matrix $H^+$ is obtained from the previous one as

$$
H^+ = H + \frac{1}{\sigma} \left( \alpha (e^t - Hw)(e^t - Hw)^T - \beta He^T (e^t - Hw) H + \tau \left( He^T (e^t - Hw) + (e^t - Hw) (e^t - Hw)^T \right) \right),
$$

(11)

where the vector $w \in \mathbb{R}^{m+n+1}$ is also an auxiliary parameter, given by

$$
w_j = \frac{1}{2} \left( \left( y^j - x^b \right)^T (x^+ - x^b) \right)^2, \quad j = 1, 2, \ldots, m,
$$

$$
w_{m+1} = 1, \quad w_{j+m+1} = \left[ x^+ - x^b \right]_j, \quad j = 1, 2, \ldots, n,
$$

(12)

with the parameters $\alpha, \beta, \tau$ and $\sigma$ assuming the values

$$
\alpha = \frac{(e^t)^T H e^t},
$$

$$
\beta = \frac{1}{2} \| x^+ - x^b \|^4 - w^T H w,
$$

$$
\tau = \frac{(e^t)^T H e^t},
$$

$$
\sigma = \alpha \beta + \tau^2.
$$

### 3.3 Update

To update the interpolation set, first we choose a point $y^t$ that will leave the set $Y_k$ to be replaced by $x^t$. Assume that the current point $x_k$ is at position $t$ in the interpolation set $Y_k$, i.e., $x_k = y^t$. The point $y^t$ that will leave $Y_k$ is such that

$$
t = \arg \max_{j \in \{1, 2, \ldots, m\}} \{ \sigma(j) \},
$$

(14)

with

$$
\sigma(j) = [H]_{jj} \left( \frac{1}{2} \| x^+ - x^b \|^4 - w^T H w \right) + ((e^j)^T H e^t)^2
$$

and the matrix $H$ and the vector $w$ defined in the previous section.

If $\sigma(t) \leq \varepsilon_1$ for a fixed tolerance $\varepsilon_1 > 0$, the set $Y_k \setminus \{ y^t \} \cup \{ x^t \}$ is considered not sufficiently posed, in the sense that we would have a very small denominator in (11). In this case, the interpolation set and the model are reconstructed as discussed in Section 3.1.

Otherwise, the interpolation set for the next iteration will be $Y_{k+1} = Y_k \setminus \{ y^t \} \cup \{ x^+ \}$. The new model $Q_{k+1}$ is defined by the parameters $c_{k+1}, g_{k+1}$ and $G_{k+1}$ given by

$$
c_{k+1} = c_k + c, \quad g_{k+1} = g_k + g \quad \text{and} \quad G_{k+1} = G_k + \sum_{j=1}^m \phi_j (y^j - x^b)(y^j - x^b)^T,$$

with

$$
\begin{bmatrix}
\varepsilon \\
c \\
g
\end{bmatrix} = (f(x^+) - Q_k(x^+))[H^+]_c.
$$

(15)

In [20], it is discussed that such model $Q_{k+1}$ is the solution of (5).
4 Numerical results

Algorithm 1 was implemented in Fortran 77. The tests were performed using the gfortran compiler (32-bits), version gcc-4.2.3, in a notebook I7, 2.1GHz, RAM of 8 Gb. The algorithm was run with input $\rho_1 = 10^{-1}$, $\varepsilon = 10^{-4}$, $\varepsilon_1 = 10^{-10}$, $\gamma = 0.1$ and $s = 10$.

The subproblems in Step 2 were solved by ALGENCAN [1, 2], a code for solving large-scale nonlinear programming problems periodically updated and publicly available at the TANGO Project web site http://www.ime.usp.br/~egbirgin/tango/. We used ALGENCAN version 2.2.1 with all parameters set to their default values, except for $\text{epsfeas} = \text{epsopt} = 10^{-8}$. This choice implies that the sequence generated by the algorithm is feasible with tolerance $10^{-8}$, in the sense that, for all $k$, $\psi(x_k) < 10^{-8}$, where $\psi: \mathbb{R}^n \to \mathbb{R}$ is the infeasibility measure given by

$$\psi(x) = \max \{||h(x)||_{\infty}, ||g^+(x)||_{\infty}\}$$

with $g^+_i(x) = \max\{0, g_i(x)\}$, for all $i = 1, \cdots, q$.

If the provided initial point is infeasible with respect to $\Omega$, we use ALGENCAN, starting on this point, to minimize the infeasibility measure $\psi^2/2$ in order to find $x_1 \in \Omega$.

We considered two variants of Algorithm 1, one that uses $m = 5$ if $n = 2$ and $m = 2n + 3$ otherwise, and another with $m = \bar{m}$ interpolation points.

To put our approach in perspective, we compared its performance with the following derivative-free algorithms found in the literature:

**HOPSPACK** (Hybrid Optimization Parallel Search Package) [19], a C++ software based on an Augmented Lagrangian approach with subproblems solved by the Generating Set Search Algorithm (GSS) [17].

**Inexact Restoration algorithm** [4], a C++ program for equality constrained problems in which each iteration is decomposed in two phases: one that uses ALGENCAN for reducing a measure of infeasibility by means of an interface between C++ and Fortran, and another one that uses GSS to minimize a suitable objective function in a linear approximation of the feasible set.

**DFO** (Derivative Free Optimization), a Fortran code based on the trust-region approach proposed in [8]. The models are constructed by quadratic interpolation using only feasible points. The subproblems are solved by the NPSOL algorithm [14].

Summing up, there are five strategies under analysis, which are:

- **S1**: Algorithm 1 with $m = 5$ if $n = 2$ and $m = 2n + 3$ otherwise;
- **S2**: Algorithm 1 with $m = \bar{m}$;
- **HP**: HOPSPACK;
- **IR**: Derivative-free inexact restoration algorithm;
- **DFO**: DFO algorithm.

For the numerical experiments, we considered the Hock-Schittkowski collection [16, 28], which is a largely used test set for optimization algorithms. Although the derivatives of all functions that define the problems are available, the collection is often used to compare the performance of derivative-free algorithms. For comparing the algorithms we used the data and performance profiles discussed in [12, 18], considering the number of objective function evaluations as the performance measure.
4.1 Performance of strategies $S_1$ and $S_2$

We tested the algorithm with all 216 problems of the Hock-Schittkowski collection [28] that involve at least one constraint beside box constraints. The dimension of the problems varies between 2 and 50 and the number of constraints between 1 and 45.

Similarly to [4], a problem is considered solved by a strategy if it found a point $x$ satisfying

$$\psi(x) < 10^{-8} \quad \text{and} \quad \frac{f(x) - f_{HS}}{\max\{1, |f(x)|, |f_{HS}|\}} \leq 10^{-4},$$

(16)

where $f_{HS}$ is the solution available in the Hock–Schittkowski collection, found by the NLPQLP code [27].

Figure 1 shows the data and performance profiles related to the number of objective function evaluations. Both strategies $S_1$ and $S_2$ were competitive in terms of robustness, solving respectively 96.3% and 97.7% of the problems. On the other hand, $S_1$ was more efficient, since strategies $S_1$ and $S_2$ performed fewer function evaluations on 86.6% and 18.1% of the problems, respectively. Furthermore, when we allowed the algorithms to perform 500 objective function evaluations, $S_1$ solved 91.5% while $S_2$ solved 84.2% of the problems. The results of this section indicate that using less than $m$ interpolation points provide better results in terms of efficiency.

![Data profile](image1)

![Performance profile](image2)

Figure 1: Data and performance profiles for the comparison between strategies $S_1$ and $S_2$.

4.2 Comparison with strategies HP and IR

In this section we compare the performance of strategies $S_1$, $S_2$, HP and IR in solving all 105 problems of the Hock-Schittkowski collection selected in [4].

As considered in [4], we say that a strategy solved a problem if it found a point $\bar{x}$ such that

$$\psi(\bar{x}) < 10^{-8} \quad \text{and} \quad \frac{f(\bar{x}) - f_{\text{min}}}{\max\{1, |f(\bar{x})|, |f_{\text{min}}|\}} \leq 0.1,$$

(17)

where $f_{\text{min}}$ is the smallest function value found among all the strategies that are being compared.

The data and performance profiles for number of function evaluations, given in Figure 2, clearly show that $S_1$ and $S_2$ were the most robust strategies. In terms of efficiency, the performance of strategy $S_1$ was superior. While $S_1$ was the best for 75.2% of the problems, strategies HP and IR were not able to solve such percentage of problems even when we allowed them to spend 50 times the number of function evaluations used by the best algorithm. Observe that, when we allowed the algorithms to perform 1000 objective function evaluations, $S_1$, $S_2$, HP and IR solved respectively 90.5%, 89.5%, 40.0% and 61.9% of the problems.
4.3 Comparison with strategy DFO

In this section we compare the performance of strategies $S_1$, $S_2$ and DFO in solving all 28 problems of the Hock-Schittkowski collection [16] selected in [8]. We adopted (17) for deciding if a strategy solved a problem.

As shown in Figure 3, DFO was more efficient than our approach in this small set of problems. On the other hand, the three strategies were similar in terms of robustness, with $S_1$ and $S_2$ presenting a slightly superior performance.

Finally, we present in Tables 1–3 some more details on the numerical results. The first four columns contain information about the problems. The labels prob, $n$, $p$, $q$ indicate, respectively, the number of the problem in the Hock–Schittkowski collection, the number of variables, the number of equality constraints and the number of inequality constraints besides bound constraints. The next columns display the minimum function value $f^*$ and the number of function evaluations $\#FE$ for each strategy. Finally, the column $f_{HS}$ displays the solution available in the Hock–Schittkowski collection. The data about solvers HP and IR were extracted from [4], while the information about DFO can be found in [8]. The symbol “-” indicates that the information is not presented in these references. When a strategy failed in solving a
the following labels were used to indicate the reason:

1. the solution does not satisfy (16);
2. the solution does not satisfy (17) considering strategies $S_1$, $S_2$, IP and IR;
3. the solution does not satisfy (17) considering the strategies $S_1$, $S_2$ and DFO;
4. the solution does not satisfy the feasibility criterion.

Conclusion. In this work, we presented a trust-region algorithm for solving constrained optimization problems in which the derivatives of the objective function are not available. The quadratic models are constructed by polynomial interpolation based on ideas proposed by Powell in [23]. The subproblems are solved by ALGENCAN [1, 2], that handles all constraints and provides only feasible iterates. The algorithm was implemented in Fortran and its performance was compared with three solvers found in the literature. The numerical results are encouraging in terms of both, efficiency and robustness.

References


Appendices
Table 1: Numerical results for the five strategies.
Table 2: Numerical results for the five strategies.
<table>
<thead>
<tr>
<th>$p$</th>
<th>$Q$</th>
<th>$T_1$</th>
<th>$S_1$</th>
<th>$T_2$</th>
<th>$S_2$</th>
<th>$T_3$</th>
<th>$S_3$</th>
<th>$T_4$</th>
<th>$S_4$</th>
<th>$T_5$</th>
<th>$S_5$</th>
<th>$T_6$</th>
<th>$S_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>25</td>
<td>185</td>
<td>1.04</td>
<td>1.06</td>
<td>1.08</td>
<td>1.10</td>
<td>1.12</td>
<td>1.14</td>
<td>1.16</td>
<td>1.18</td>
<td>1.20</td>
<td>1.22</td>
<td>1.24</td>
</tr>
<tr>
<td>30</td>
<td>24</td>
<td>184</td>
<td>1.05</td>
<td>1.07</td>
<td>1.09</td>
<td>1.11</td>
<td>1.13</td>
<td>1.15</td>
<td>1.17</td>
<td>1.19</td>
<td>1.21</td>
<td>1.23</td>
<td>1.25</td>
</tr>
<tr>
<td>31</td>
<td>23</td>
<td>183</td>
<td>1.06</td>
<td>1.08</td>
<td>1.10</td>
<td>1.12</td>
<td>1.14</td>
<td>1.16</td>
<td>1.18</td>
<td>1.20</td>
<td>1.22</td>
<td>1.24</td>
<td>1.26</td>
</tr>
<tr>
<td>32</td>
<td>22</td>
<td>182</td>
<td>1.07</td>
<td>1.09</td>
<td>1.11</td>
<td>1.13</td>
<td>1.15</td>
<td>1.17</td>
<td>1.19</td>
<td>1.21</td>
<td>1.23</td>
<td>1.25</td>
<td>1.27</td>
</tr>
<tr>
<td>33</td>
<td>21</td>
<td>181</td>
<td>1.08</td>
<td>1.10</td>
<td>1.12</td>
<td>1.14</td>
<td>1.16</td>
<td>1.18</td>
<td>1.20</td>
<td>1.22</td>
<td>1.24</td>
<td>1.26</td>
<td>1.28</td>
</tr>
</tbody>
</table>

Table 3: Remaining results for strategies $S_1$ and $S_2$. 