Large-scale unconstrained optimization using separable cubic modeling and matrix-free subspace minimization*

C. P. Brás† J. M. Martínez‡ M. Raydan§

July 31, 2019

Abstract

We present a new algorithm for solving large-scale unconstrained optimization problems that uses cubic models, matrix-free subspace minimization, and secant-type parameters for defining the cubic terms. We also propose and analyze a specialized trust-region strategy to minimize the cubic model on a properly chosen low-dimensional subspace, which is built at each iteration using the Lanczos process. For the convergence analysis we present, as a general framework, a model trust-region subspace algorithm with variable metric and we establish asymptotic as well as complexity convergence results. Preliminary numerical results, on some test functions and also on the well-known disk packing problem, are presented to illustrate the performance of the proposed scheme when solving large-scale problems.

Keywords: Smooth unconstrained minimization, cubic modeling, subspace minimization, trust-region strategies, Newton-type methods, Lanczos method, disk packing problem.

1 Introduction

We consider the unconstrained minimization problem

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a sufficiently smooth function of the variables $x \in \mathbb{R}^n$. We are specially interested in the case in which n is very large.

Methods for solving this class of problems usually rely on gradient-type iterations, which are effective when one has a single global minimizer or a global minimizer with a sufficiently large basin of attraction. In the presence of saddle points, perhaps on the boundary of a

*This work was supported by PRONEX-CNPq/FAPERJ (E-26/111.449/2010-APQ1), CEPID–Industrial Mathematics/FAPESP (Grant 2011/51305-02), FAPESP (projects 2013/05475-7 and 2013/07375-0), and by the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) through the project UID/MAT/00297/2019 (CMA).

†Centro de Matemática e Aplicações (CMA), FCT, UNL, and Departamento de Matemática, FCT, UNL, 2829-516 Caparica, Portugal. (mb@fct.unl.pt).

‡Department of Applied Mathematics, IMECC-UNICAMP, University of Campinas, Rua Sérgio Buarque de Holanda, 651 Cidade Universitária "Zeferino Vaz", Distrito Barão Geraldo, 13083-859 Campinas SP, Brazil. (martinez@ime.unicamp.br).

§Centro de Matemática e Aplicações (CMA), FCT, UNL, 2829-516 Caparica, Portugal. (m.raydan@fct.unl.pt).

very narrow region on which the function decreases, gradient methods tend to converge to stationary points that are far from being global solutions. Standard techniques to avoid that situation involve variations of Newton's method, trust-region strategies, and negative-curvature directions; see, e.g., [14, 15, 16, 24, 25, 26, 27, 30, 34, 35]. The variable norm trust-region strategy, proposed in [24], minimizes the standard quadratic Taylor model plus a separable cubic approximation of third order derivatives, and for that it requires a full eigenvalue decomposition (Schur factorization) of the (approximate) Hessian matrix at every iteration. Unfortunately, when the number of variables is large, the employment of pure Newton ideas is prohibitive. As a consequence, strategies as the one defined in [24], which seems to be effective for small and medium-scale problems, cannot be applied for large-scale problems.

A possible remedy is to use subspace optimization, in which at each iteration a model of the objective function is minimized on a properly chosen (usually low-dimensional) subspace; see, e.g., [7, 8, 34]. In the case of Newton-type optimization schemes for large scale nonconvex problems, building the low-dimensional subspace via Lanczos method has several advantages; see, e.g., [4, 7, 8, 14, 15, 16, 27]. In our context, the main advantage is the tendency to find eigenvectors associated to extreme eigenvalues, so that, in general, negative curvature directions are detected very quickly.

In this work, combining Lanczos method with the separable cubic model in [24], we devise an affordable matrix-free method for large-scale problems that converges to points where the gradient vanishes and, very likely, the Hessian is positive semi-definite. The main advantage of the new approach is that the required step per Newton-type iteration is obtained in a low-dimensional subspace, in sharp contrast with the scheme proposed in [24] for which it is computed on the entire large-dimensional space. Moreover, the arguments and the numerical experiments in [24] reveal that the new strategy has good chances of avoiding local-nonglobal minimizers.

The remainder of this article is structured as follows: in Section 2 we describe the algorithm and analyze its convergence and complexity. In Section 3 a detailed description of the separable cubic model algorithm is presented as well as implementation details. Numerical results are reported in Section 4 to illustrate the behavior of the proposed scheme when solving large-scale problems for which function evaluations are as expensive as gradient and Hessian evaluations. Final remarks are stated in Section 5.

2 Main algorithm, convergence, and complexity

The algorithm that will be presented in this section is of trust-region type. At each iteration the algorithm minimizes a model of the objective function subject to a trust region defined by a norm that varies from iteration to iteration. The reasons for this variation will become apparent in the next section and coincide with the ones analyzed in our previous paper [24]. In addition to the restriction on the size of the trust region, the new iterate is forced to belong to a (small dimensional) affine subspace. In this section we will describe the algorithm in minimal terms and we will deduce convergence and complexity results. A special model with cubic terms, including implementation details, will be discussed in Section 3.

Let $c_{max} > 1$ and $0 < c_{min} < 1$. Throughout this section we will denote by $\mathcal{F}_{c_{min},c_{max}}$ the set of norms on $\mathbb{R}^n$ such that, for all $\|\cdot\| \in \mathcal{F}_{c_{min},c_{max}}$ and $z \in \mathbb{R}^n$,

$$c_{min}\|z\|_2 \leq \|z\| \leq c_{max}\|z\|_2. \tag{2}$$

Obviously, $\|\cdot\|_2 \in \mathcal{F}_{c_{min},c_{max}}$.

We will assume that there exists $\gamma > 0$ such that, for all $x, s \in \mathbb{R}^n$ and $\|\cdot\| \in \mathcal{F}_{c_{min},c_{max}}$,

$$f(x+s) \leq f(x) + \nabla f(x)^T s + \gamma \|s\|^2. \tag{3}$$

Assumption (3) holds if the gradient satisfies a Lipschitz condition on $\mathbb{R}^n$. The Lipschitz condition on the gradient, however, is not explicitly used in our context.

The employment of different norms in the context of trust-region methods has been studied in [9]. In [24] trust-region methods with high-order models were introduced employing variable norms for proving convergence properties. In [1] and [2] a variable-norm approach was used in order to introduce line-search Newtonian methods with optimal complexity.

For all $x \in \mathbb{R}^n$, we will assume that $M(x,.)$ is such that there exists $\beta > 0$ such that for all $s \in \mathbb{R}^n$, and $\|\cdot\| \in \mathcal{F}_{c_{min},c_{max}}$,

$$|M(x,s) - \nabla f(x)^T s| \leq \beta \|s\|^2. \tag{4}$$

We will say that $M(x,s)$ is *a (first-order) model* for $f(x+s) - f(x)$. All the results that lead to Theorems 2.1 and 2.2 assume that $M(x,s)$ is a first-order model. For proving Theorem 2.3, however, we will use that $M(x,s)$ is a second-order model of $f(x+s)-f(x)$. In that case, besides (4) we will assume that there exists $\beta_2 > 0$ such that, for all $x, s \in \mathbb{R}^n$ and $\|\cdot\| \in \mathcal{F}_{c_{min},c_{max}}$,

$$\left| M(x,s) - \nabla f(x)^T s - \frac{1}{2} s^T \nabla^2 f(x) s \right| \leq \beta_2 \|s\|^3. \tag{5}$$

By elementary calculus, (4) implies that $\nabla M(x,0) = \nabla f(x)$ and (5) implies that $\nabla^2 M(x,0) = \nabla^2 f(x)$.

Lemma 2.1 *Whenever $\|\cdot\| \in \mathcal{F}_{c_{min},c_{max}}$, if $x, s \in \mathbb{R}^n$, we have:*

$$f(x+s) - f(x) \leq M(x,s) + (\beta + \gamma)\|s\|^2.$$

Proof. By (3) and (4) we have:

$$\begin{aligned} f(x+s) - f(x) &\leq \nabla f(x)^T s + \gamma \|s\|^2 \\ &= M(x,s) - M(x,s) + \nabla f(x)^T s + \gamma \|s\|^2 \\ &\leq M(x,s) + |-M(x,s) + \nabla f(x)^T s| + \gamma \|s\|^2 \\ &\leq M(x,s) + (\beta + \gamma)\|s\|^2. \end{aligned}$$

as we wanted to prove. □

We now present our model trust-region algorithm.

Algorithm 2.1

Let $\alpha \in (0, 1/2]$, $\eta \in (0, 1/2)$, and $\delta_{max} > 0$ be algorithmic parameters. Assume that $x_0 \in \mathbb{R}^n$ is a given initial approximation to the solution. Set $k \leftarrow 0$.

Step 1: **decide** whether to finish the execution of the algorithm at x_k or not according to some stopping criterion to be defined later.

Step 2: **choose** $\delta = \delta_{zero,k} \in (0, \delta_{max}]$, $S_k \subseteq \mathbb{R}^n$, a subspace of $\mathbb{R}^n$ such that $\nabla f(x_k) \in S_k$, and a norm $\|\cdot\|_{x_k} \in \mathcal{F}_{c_{min},c_{max}}$.

Step 3: **minimize**, with respect to $s \in \mathbb{R}^n$, the model $M(x_k, s)$ subject to $s \in S_k$ and $\|s\|_{x_k} \leq \delta$, obtaining a (global) minimizer s_{trial} for this subproblem.

Step 4: **test** the sufficient descent condition

$$f(x_k + s_{trial}) \leq f(x_k) + \alpha M(x_k, s_{trial}). \tag{6}$$

if (6) is fulfilled, **define** $\delta_k = \delta$, $s_k = s_{trial}$, $x_{k+1} = x_k + s_k$, **update** $k \leftarrow k + 1$ and go to **Step 1**.

else **define** $\delta_{new} \in [\eta\delta, (1 - \eta)\delta]$, **update** $\delta \leftarrow \delta_{new}$ and go to **Step 3**.

end if

In Section 3 we will describe a practical implementation of Algorithm 2.1. In this implementation the norm $\|\cdot\|_{x_k}$ will be chosen according to the scheme that follows. Suppose that the dimension of S_k is $n_k \ll n$, $\{w_1, \ldots, w_n\}$ is an orthonormal basis of $\mathbb{R}^n$, and $\{w_1, \ldots, w_{n_k}\}$ is an orthonormal basis of S_k. Then, for all $x \in \mathbb{R}^n$ we have that $x = \sum_{i=1}^{n} (x^T w_i) w_i$ and $\|\cdot\|_{x_k}$ will be the sup-norm corresponding to this basis. In other words, for all $x \in \mathbb{R}^n$, $\|x\|_{x_k} = \max\{|x^T w_1|, \ldots, |x^T w_n|\}$. Since $\|x\|_2 = \sqrt{\sum_{i=1}^{n} (x^T w_i)^2}$ we clearly have that (2) holds with c_{min} and c_{max} independent of k. The way in which S_k and the orthonormal basis are computed will be given in Section 3.

This section will be devoted to prove convergence and complexity properties of Algorithm 2.1. Since this algorithm is a trust-region method in which the approximating model (not necessarily quadratic) coincides with the objective function up to first-order terms, it is not surprising that convergence to first-order stationary points occurs and stationary points with precision ε are obtained employing at most $O(\varepsilon^{-2})$ function and derivative evaluations. A non-standard trust-region method with cubic regularization ingredients was introduced in [12]. Our algorithm does not use (explicit or implicit) cubic regularization [17, 19, 23, 28], so that we cannot expect to obtain first-order complexity better than $O(\varepsilon^{-2})$, as in the methods analyzed in [18]. The example showed in [6] may be trivially adapted to show that such complexity result is sharp. Moreover, the number of iterations k such that the subspace S_k contains some negative-curvature direction is $O(\varepsilon^{-3})$ as in cubic-regularization methods.

In what follows, a sequence of lemmas will lead to proving Theorems 2.1, 2.2, and 2.3. Theorem 2.1 states that, due to sufficient descent conditions proved in the lemmas, as these conditions depend on $\|\nabla f(x_k)\|$, the norm of $\nabla f(x_k)$ must tend to zero. Using the same type of arguments, Theorem 2.2 computes the maximal number of iterations and evaluations that may occur with the norm of the gradient being greater than a given tolerance. Theorem 2.3 is interesting because it gives a hint on the difference between our method and the steepest descent approach. Essentially, this theorem says that the number of iterations at which S_k contains a sufficiently negative-curvature direction defined by $\epsilon_2 > 0$ is bounded above by a multiple of ϵ_2^{-3}. This property indicates that, if we want convergence to local minimizers, we must try to compute subspaces that contain negative curvature directions, when they exist. This is the observation that leads to our choices of S_k. As in the case of Theorems 2.1 and 2.2, the proof of Theorem 2.3 rests on sufficient descent properties proved in previous lemmas.

Lemma 2.2 quantifies the decrease exhibited by the model $M(x_k, s)$ from an iterate x_k at which the gradient of the objective function does not vanish.

Lemma 2.2 *Let x_k be a generic iterate of Algorithm 2.1 at which the algorithm does not stop and such that $\|\nabla f(x_k)\|_2 > \varepsilon > 0$. Let s_{trial} be defined as in Step 3 of the algorithm. Then, if $\delta \geq \frac{\varepsilon c_{min}^2}{2\beta c_{max}}$, we obtain that*

$$M(x_k, s_{trial}) \leq -\frac{\varepsilon^2 c_{min}^2}{4\beta c_{max}^2} < 0,$$

whereas, if $\delta < \frac{\varepsilon c_{min}^2}{2\beta c_{max}}$, we have that $M(x_k, s_{trial}) \leq \varphi(\delta) \leq -\frac{\varepsilon\delta}{2c_{max}} < 0$, where

$$\varphi(\delta) \equiv -\varepsilon\delta/c_{max} + \beta\delta^2/c_{min}^2. \tag{7}$$

Proof. By $\|\nabla f(x_k)\|_2 > \varepsilon$ and (2),

$$\|\nabla f(x_k)\|_{x_k} \geq c_{min}\|\nabla f(x_k)\|_2 > c_{min}\varepsilon. \tag{8}$$

Define, for all $t \in (0, 1]$, $s(t) = -t\nabla f(x_k)\delta/\|\nabla f(x_k)\|_{x_k}$. Then, $s(t) \in S_k$ and

$$\|s(t)\|_{x_k} = t\delta \leq \delta. \tag{9}$$

Therefore, $s(t)$ is a feasible point of the subproblem defined at Step 3. Therefore, by (4),

$$M(x_k, s_{trial}) \leq M(x_k, s(t)) \leq \nabla f(x_k)^T s(t) + \beta\|s(t)\|_2^2.$$

Thus, by the definition of $s(t)$ and (2),

$$M(x_k, s_{trial}) \leq -\frac{t\nabla f(x_k)^T\nabla f(x_k)\delta}{\|\nabla f(x_k)\|_{x_k}} + \beta\|s(t)\|_2^2 \leq -\frac{t\nabla f(x_k)^T\nabla f(x_k)\delta}{c_{max}\varepsilon} + \beta\|s(t)\|_2^2.$$

So, by (2) and $\|\nabla f(x_k)\|_2 \geq \varepsilon$, we have that $M(x_k, s_{trial}) \leq -t\varepsilon\delta/c_{max} + \beta\|s(t)\|_{x_k}^2/c_{min}^2$. Thus, by the definition of $s(t)$,

$$M(x_k, s_{trial}) \leq -t\varepsilon\delta/c_{max} + \beta t^2\delta^2/c_{min}^2. \tag{10}$$

The unconstrained minimizer t_{unc}, with respect to t, of $-t\varepsilon\delta/c_{max} + \beta t^2\delta^2/c_{min}^2$ is given by

$$-\varepsilon\delta/c_{max} + 2t_{unc}\beta\delta^2/c_{min}^2 = 0.$$

Hence,

$$t_{unc} = \frac{\varepsilon\delta/c_{max}}{2\beta\delta^2/c_{min}^2} = \frac{\varepsilon c_{min}^2}{2\beta\delta c_{max}}.$$

If $t_{unc} \leq 1$ the minimum value of $-t\varepsilon\delta/c_{max} + \beta t^2\delta^2/c_{min}^2$ for $t \leq 1$ takes place when $t = t_{unc}$ and, when $t_{unc} \geq 1$ that minimum value occurs when $t = 1$.

Now, $t_{unc} \leq 1$ if and only if $\frac{\varepsilon c_{min}^2}{2\beta\delta c_{max}} \leq 1$ which is equivalent to

$$\delta \geq \frac{\varepsilon c_{min}^2}{2\beta c_{max}}. \tag{11}$$

Then, by (10), when (11) takes place, we have that:

$$M(x_k, s_{trial}) \leq -t_{unc}\varepsilon\delta/c_{max} + \beta t_{unc}^2\delta^2/c_{min}^2 = -\frac{\varepsilon c_{min}^2}{2\beta\delta c_{max}}\varepsilon\delta/c_{max} + \beta[\frac{\varepsilon c_{min}^2}{2\beta\delta c_{max}}]^2\delta^2/c_{min}^2$$

$$= \frac{c_{min}^2}{c_{max}^2}\left[-\frac{\varepsilon^2}{2\beta} + \beta[\frac{\varepsilon}{2\beta}]^2\right] = -\frac{\varepsilon^2 c_{min}^2}{4\beta c_{max}^2}.$$

On the other hand, when $\delta \leq \frac{\varepsilon c_{min}^2}{2\beta c_{max}}$ and, consequently, $t_{unc} \geq 1$, the minimum of $-t\varepsilon\delta/c_{max} + \beta t^2\delta^2/c_{min}^2$ occurs when $t = 1$. Then, by (10), the minimum of $\varphi(\delta)$ for $\delta \leq \frac{\varepsilon c_{min}^2}{2\beta c_{max}}$ occurs at $\delta = \frac{\varepsilon c_{min}^2}{2\beta c_{max}}$.

Note that $\varphi(0) = 0$ and $\varphi'(0) = -\varepsilon/c_{max}$, so $\varphi(\delta)$ is a decreasing parabola between 0 and $\frac{\varepsilon c_{min}^2}{2\beta c_{max}}$ with minimum $-\frac{\varepsilon^2 c_{min}^2}{4\beta c_{max}^2}$ at $\frac{\varepsilon c_{min}^2}{2\beta c_{max}}$. Therefore, for $\delta \in [0, \frac{\varepsilon c_{min}^2}{2\beta c_{max}}]$, $\varphi(\delta)$ is bounded above by the line that joins the point $(0, 0)$ with the point $(\frac{\varepsilon c_{min}^2}{2\beta c_{max}}, -\frac{\varepsilon^2 c_{min}^2}{4\beta c_{max}^2})$. This completes the proof. □

The following lemma ensures that, when the gradient at x_k is not null, the sufficient descent condition (6) is verified if the trust-region radius δ is small enough. Therefore, iteration k is well defined when $\nabla f(x_k) \neq 0$.

Lemma 2.3 *Let x_k be a generic iterate of Algorithm 2.1 at which the algorithm does not stop, such that $\|\nabla f(x_k)\|_2 > \varepsilon > 0$. Let s_{trial} be defined as in Step 3 of the algorithm. Then, if*

$$\frac{\varepsilon c_{min}^2}{2\beta c_{max}} \leq \delta \leq \sqrt{\frac{1-\alpha}{4(\beta+\gamma)\beta}}\, c_{min}\varepsilon \tag{12}$$

or

$$\delta < \frac{\varepsilon c_{min}^2}{2\beta c_{max}} \text{ and } \delta \leq \frac{c_{min}^2(1-\alpha)\varepsilon}{(\beta+\gamma)c_{min}^2 c_{max} + (1-\alpha)\beta c_{max}}, \tag{13}$$

the sufficient descent condition (6) takes place. Moreover, condition (6) is fulfilled after a finite number of reductions of δ.

Proof. By Lemmas 2.1 and 2.2, and (2), since $M(x_k, s_{trial}) < 0$ and $\|s_{trial}\|_{x_k} \leq \delta$, we have:

$$\frac{f(x_k + s_{trial}) - f(x_k)}{M(x_k, s_{trial})} \geq 1 + \frac{(\beta+\gamma)\delta^2}{M(x_k, s_{trial})}. \tag{14}$$

By Lemma 2.2 and (14), when $\delta \geq \frac{\varepsilon c_{min}^2}{2\beta c_{max}}$ we have that

$$\frac{f(x_k + s_{trial}) - f(x_k)}{M(x_k, s_{trial})} \geq 1 - \frac{(\beta+\gamma)4\beta c_{max}^2\delta^2}{c_{min}^2\varepsilon^2}$$

and, by Lemma 2.2 and (7), when $\delta < \frac{\varepsilon c_{min}^2}{2\beta c_{max}}$ we have that

$$\frac{f(x_k + s_{trial}) - f(x_k)}{M(x_k, s_{trial})} \geq 1 + \frac{(\beta+\gamma)\delta^2}{\varphi(\delta)} = 1 + \frac{(\beta+\gamma)\delta c_{max} c_{min}^2}{-\varepsilon c_{min}^2 + \beta\delta c_{max}}.$$

But, when $\delta < \frac{\varepsilon c_{min}^2}{2\beta c_{max}}$ one has that $2\beta\delta c_{max} - \varepsilon c_{min}^2 < 0$ and, so, $\varepsilon c_{min}^2 - \beta\delta c_{max} > 0$ and

$$\frac{f(x_k + s_{trial}) - f(x_k)}{M(x_k, s_{trial})} \geq 1 - \frac{(\beta+\gamma)\delta c_{min}^2 c_{max}}{\varepsilon c_{min}^2 - \beta\delta c_{max}} < 1.$$

Therefore, if

$$\delta \geq \frac{\varepsilon c_{min}^2}{2\beta c_{max}} \text{ and } \frac{(\beta+\gamma)4\beta c_{max}^2\delta^2}{\varepsilon^2 c_{min}^2} \leq 1 - \alpha \tag{15}$$

or

$$\delta < \frac{\varepsilon c_{min}^2}{2\beta c_{max}} \text{ and } \frac{(\beta+\gamma)\delta c_{min}^2 c_{max}}{\varepsilon c_{min}^2 - \beta\delta c_{max}} \leq 1-\alpha, \tag{16}$$

it follows that

$$\frac{f(x_k + s_{trial}) - f(x)}{M(x_k, s_{trial})} \geq \alpha. \tag{17}$$

By (15) and (16), if

$$\delta \geq \frac{\varepsilon c_{min}^2}{2\beta c_{max}} \text{ and } \delta \leq \sqrt{\frac{1-\alpha}{4(\beta+\gamma)\beta}} \frac{c_{min}}{c_{max}} \varepsilon$$

or

$$\delta < \frac{\varepsilon c_{min}^2}{2\beta c_{max}} \text{ and } \delta \leq \frac{c_{min}^2 (1-\alpha)\varepsilon}{(\beta+\gamma)c_{min}^2 c_{max} + (1-\alpha)\beta c_{max}},$$

we have that (17) takes place. Since (17) is equivalent to (6), the first part of the proof is complete.

Now, independently of the first choice of δ at iteration k, successive choices of δ satisfy $\delta_{new} \leq (1-\eta)\delta$, where $0 < 1-\eta < 1$. Therefore, after a finite number of reductions of δ at least one of the conditions (12) or (13) is satisfied. This completes the proof. □

In Lemma 2.4 we quantify the decrease obtained in the objective function, at an iteration where the gradient does not vanish, when the conditions on δ established in Lemma 2.4 are verified.

Lemma 2.4 *Let x_k be a generic iterate of Algorithm 2.1 at which the algorithm does not stop and such that $\|\nabla f(x_k)\|_2 > \varepsilon > 0$. Let s_{trial} be defined as in Step 3 of the algorithm. Then, if one of the conditions (12) or (13) takes place the algorithm accepts the trust-region radius δ, defines $s_k = s_{trial}$, and, in the case of (12),*

$$f(x_{k+1}) = f(x_k + s_k) \leq f(x_k) - \alpha \frac{\varepsilon^2}{4\beta} \frac{c_{min}^2}{c_{max}^2}, \tag{18}$$

while, in the case of (13),

$$f(x_{k+1}) = f(x_k + s_k) \leq f(x_k) + \alpha\varphi(\delta) < f(x_k), \tag{19}$$

where $\varphi(\delta)$ is defined by (7).

Proof. By Lemma 2.3 both in the cases (12) and (13) the step s_{trial} is accepted and the condition (6) is satisfied. In the case of (12), by (6) and Lemma 2.2, we have:

$$f(x_{k+1}) = f(x_k + s_{trial}) \leq f(x_k) + \alpha M(x_k, s_{trial}) \leq f(x_k) - \alpha \frac{\varepsilon^2}{4\beta} \frac{c_{min}^2}{c_{max}^2}.$$

In the case (13), also by (6) and Lemma 2.2, we have:

$$\begin{aligned} f(x_{k+1}) = f(x_k + s_{trial}) &\leq f(x_k) + \alpha M(x_k, s_{trial}) \\ &\leq f(x_k) + \alpha\varphi(\delta) < f(x_k). \end{aligned}$$

This completes the proof. □

In the following lemma we give a lower bound on the size of the accepted trust-region radius δ_k at an iteration where $\nabla f(x_k)$ is not null.

Lemma 2.5 *Let x_k be a generic iterate of Algorithm 2.1 at which the algorithm does not stop and such that $\|\nabla f(x_k)\|_2 > \varepsilon > 0$. Let s_{trial} be defined as in Step 3 of the algorithm. Then, the accepted trust-region radius δ_k satisfies:*

$$\delta_k \geq \min\left\{\delta_{zero,k}, \eta\frac{\varepsilon c_{min}^2}{2\beta c_{max}}, \frac{\eta c_{min}^2(1-\alpha)\varepsilon}{(\beta+\gamma)c_{min}^2 c_{max} + (1-\alpha)\beta c_{max}}\right\}.$$

Proof. If the accepted trust-region radius δ_k is equal to $\delta_{zero,k}$ we are done. Therefore, we only need to analyze the case in which $\delta_k < \delta_{zero,k}$. In that case, there exists a trust-region radius $\delta_{previous}$ tested immediately before δ_k such that $\delta_{previous}$ was rejected by the test (6). By construction, $\delta_{previous} \geq \frac{\delta_k}{1-\eta}$ and $\delta_{previous} \leq \frac{\delta_k}{\eta}$. Now, since $\delta_{previous}$ was rejected, by Lemma 2.3, it turns out that

$$\delta_{previous} \geq \frac{\varepsilon c_{min}^2}{2\beta c_{max}} \text{ or } \delta_{previous} > \frac{c_{min}^2(1-\alpha)\varepsilon}{(\beta+\gamma)c_{min}^2 c_{max} + (1-\alpha)\beta c_{max}}.$$

Therefore, since $\delta_{previous} \leq \frac{\delta_k}{\eta}$, we have that

$$\delta_k \geq \eta\frac{\varepsilon c_{min}^2}{2\beta c_{max}} \text{ or } \delta_k > \frac{\eta c_{min}^2(1-\alpha)\varepsilon}{(\beta+\gamma)c_{min}^2 c_{max} + (1-\alpha)\beta c_{max}}.$$

This completes the proof. □

In Lemma 2.6 we express the decrease in the objective function, at an iteration in which the gradient is not null, as a function of the gradient size and the initial trust-region radius chosen at Step 2.

Lemma 2.6 *Let x_k be a generic iterate of Algorithm 2.1 at which the algorithm does not stop and such that $\|\nabla f(x_k)\|_2 > \varepsilon > 0$. Let s_{trial} be defined as in Step 3 of the algorithm. Then,*

$$f(x_{k+1}) \leq f(x_k) - \min\{\kappa_1\delta_{zero,k}\varepsilon,\ \kappa_2\varepsilon^2\},$$

where

$$\kappa_1 = \frac{\alpha}{2c_{max}}, \tag{20}$$

and

$$\kappa_2 = \min\left\{\frac{\eta\alpha c_{min}^2}{4\beta c_{max}^2}, \frac{c_{min}^2\eta\alpha(1-\alpha)}{2(\beta+\gamma)c_{min}^2 c_{max}^2 + 2(1-\alpha)\beta c_{max}^2}\right\}. \tag{21}$$

Proof. By (6) we have that

$$f(x_{k+1}) \leq f(x_k) + \alpha M(x_k, s_k).$$

Then, by Lemma 2.4, if $\delta_k \geq \frac{\varepsilon c_{min}^2}{2\beta c_{max}}$ we have that

$$f(x_{k+1}) \leq f(x_k) - \alpha\frac{\varepsilon^2 c_{min}^2}{4\beta c_{max}^2}, \tag{22}$$

while, if $\delta_k < \frac{\varepsilon c_{min}^2}{2\beta c_{max}}$, by Lemma 2.2, we obtain

$$f(x_{k+1}) \leq f(x_k) - \alpha\frac{\varepsilon}{2c_{max}}\delta_k.$$

So, by Lemma 2.5, when $\delta_k < \frac{\varepsilon c_{min}^2}{2\beta c_{max}}$ it follows that

$$f(x_{k+1}) \leq f(x_k) - \alpha \frac{\varepsilon}{2c_{max}} \min\left\{\delta_{zero,k}, \eta \frac{\varepsilon c_{min}^2}{2\beta c_{max}}, \eta \frac{c_{min}^2(1-\alpha)\varepsilon}{(\beta+\gamma)c_{min}^2 c_{max} + (1-\alpha)\beta c_{max}}\right\}. \quad (23)$$

Then, by (22) and (23),

$$f(x_{k+1}) \leq f(x_k) - \min\left\{\frac{\alpha\varepsilon\delta_{zero,k}}{2c_{max}}, \eta\alpha \frac{\varepsilon^2 c_{min}^2}{4\beta c_{max}^2}, \eta \frac{c_{min}^2 \alpha(1-\alpha)\varepsilon^2}{2(\beta+\gamma)c_{min}^2 c_{max}^2 + 2(1-\alpha)\beta c_{max}^2}\right\}. \quad (24)$$

Then, by (22) and (24),

$$f(x_{k+1}) \leq f(x_k) - \min\left\{\frac{\alpha\varepsilon\delta_{zero,k}}{2c_{max}}, \min\left\{\alpha \frac{\varepsilon^2 c_{min}^2}{4\beta c_{max}^2}, \eta\alpha \frac{\varepsilon^2 c_{min}^2}{4\beta c_{max}^2}, \eta \frac{c_{min}^2 \alpha(1-\alpha)\varepsilon^2}{c_{max}^2\,(2(\beta+\gamma)c_{min}^2 + 2(1-\alpha)\beta)}\right\}\right\}.$$

Since $\alpha < 1$, this implies the desired result. □

The following theorem gives an asymptotic result. We run the algorithm stopping only when the gradient at the current iterate vanishes and we consider two different situations with respect to the trust-region radius choice at the beginning of each iteration. We will show that the gradient tends to zero in both situations. As a consequence, limit points of sequences generated by Algorithm 2.1 are stationary.

Theorem 2.1 *Assume that we run Algorithm 2.1, stopping only when $\nabla f(x_k) = 0$ at Step 1, in one of the following versions:*

1. *For all $k \in \mathbb{N}$, the initial trust-region radius, chosen at Step 2, is such that $\delta_{zero,k} \geq \delta_{\min} > 0$, where $\delta_{\min}$ is given independently of k.*
2. *For all $k \in \mathbb{N}$, the initial trust-region, chosen at Step 2, is such that $\delta_{zero,k} \geq \kappa_3 \|\nabla f(x_k)\|_2 > 0$, where $\kappa_3 > 0$ is given independently of k.*

Let the objective function f be bounded below. Then, either the algorithm stops at some k with $\nabla f(x_k) = 0$ or

$$\lim_{k\to\infty} \nabla f(x_k) = 0.$$

Moreover, if x_* is a limit point of the sequence $\{x_k\}$, generated by Algorithm 2.1, we have that $\nabla f(x_*) = 0$.

Proof. Let us call $\varepsilon_k = \|\nabla f(x_k)\|/2$. By Lemma 2.6, we have that for all $k = 0, 1, 2, \ldots$,

$$f(x_{k+1}) \leq f(x_k) - \min\{\kappa_1 \delta_{zero,k}\varepsilon_k,\ \kappa_2 \varepsilon_k^2\}.$$

Therefore, for the first version of the algorithm, as $\delta_{zero,k} \geq \delta_{\min}$, one has:

$$f(x_{k+1}) \leq f(x_k) - \min\{\kappa_1 \delta_{\min}\varepsilon_k,\ \kappa_2 \varepsilon_k^2\}$$

whereas, for the second version, as $\delta_{zero,k} \geq \kappa_3 \|\nabla f(x_k)\|_2$,

$$f(x_{k+1}) \leq f(x_k) - \min\{2\kappa_3\kappa_1 \varepsilon_k^2,\ \kappa_2 \varepsilon_k^2\}.$$

In both cases, if the sequence does not stop at a final k, since $\{f(x_k)\}$ is bounded below, we must have $\lim_{k\to\infty} \varepsilon_k = 0$. Therefore, $\lim_{k\to\infty} \nabla f(x_k) = 0$.

If x_* is a limit point, we have that $\nabla f(x_k)$ tends to zero along a subsequence that tends to x_*. Then, by the continuity of ∇f, we have that $\nabla f(x_*) = 0$. □

The following is a complexity result. We assume that the algorithm stops when the size of the gradient is smaller than a tolerance ϵ_1. Consequently, we will prove that the number of iterations, functional, and derivative evaluations, that are necessary for such objective is bounded above by a multiple of ϵ_1^{-2}.

Theorem 2.2 *Assume that we run Algorithm 2.1, stopping only if $\|\nabla f(x_k)\|_2 \leq \epsilon_1 > 0$ and that there exist $\kappa_3 > 0$ such that $\delta_{zero,k} \geq \kappa_3 \|\nabla f(x_k)\|_2$ for all $k = 0, 1, 2, \ldots$. Then, if f_{min} is a lower bound of f onto $\mathbb{R}^n$, the algorithm stops satisfying $\|\nabla f(x_k)\|_2 \leq \epsilon_1$ employing at most $\left\lfloor \frac{f(x_0)-f_{min}}{\kappa \epsilon_1^2} \right\rfloor$ iterations, where $\kappa = \min\{\kappa_1\kappa_3, \kappa_2\}$. Moreover, if $\kappa_4 > 0$ is such that $\delta_{zero,k} \leq \kappa_4 \|\nabla f(x_k)\|_2$ for all $k = 0, 1, 2, \ldots$, the total number of function evaluations does not exceed $\left\lfloor \frac{n_1(f(x_0)-f_{min})}{\kappa \epsilon_1^2} + 1 \right\rfloor$, where n_1 is the first integer that verifies*

$$(1-\eta)^{n_1}\kappa_4 \leq \min\left\{ \frac{c_{min}^2}{4\beta c_{max}}, \frac{c_{min}^2(1-\alpha)}{2(\beta+\gamma)c_{min}^2 c_{max} + 2(1-\alpha)\beta c_{max}} \right\}. \tag{25}$$

Proof. Let x_k be such that $\|\nabla f(x_k)\|_2 > \epsilon_1$. Define $\varepsilon_k = (\|\nabla f(x_k)\|_2 + \epsilon_1)/2$. Therefore, by Lemma 2.6 and the choice of the first δ at iteration k,

$$f(x_{k+1}) \leq f(x_k) - \min\{\kappa_1\kappa_3\varepsilon_k^2,\ \kappa_2\varepsilon_k^2\},$$

where $\kappa_1 > 0$ and $\kappa_2 > 0$ are given in (20) and (21). Therefore,

$$f(x_{k+1}) \leq f(x_k) - \kappa\epsilon_1^2,$$

where $\kappa = \min\{\kappa_1\kappa_3, \kappa_2\}$. Then, for all $k = 1, 2, \ldots$ such that $\|\nabla f(x_k)\|_2 > \epsilon_1$,

$$f(x_k) \leq f(x_0) - k\kappa\epsilon_1^2.$$

Therefore,

$$k \leq \frac{f(x_0) - f(x_k)}{\kappa\epsilon_1^2} \leq \frac{f(x_0) - f_{\min}}{\kappa\epsilon_1^2}.$$

So, the first part of the theorem is proved. It remains to prove that the number of reductions of δ per iteration is bounded.

Recall that n_1 is the first integer such that (25) holds.

Then,

$$(1-\eta)^{n_1}\kappa_4\|\nabla f(x_k)\|_2 \leq \frac{c_{min}^2}{4\beta c_{max}}\|\nabla f(x_k)\|_2. \tag{26}$$

and

$$(1-\eta)^{n_1}\kappa_4\|\nabla f(x_k)\|_2 \leq \frac{c_{min}^2(1-\alpha)}{2(\beta+\gamma)c_{min}^2 c_{max} + 2(1-\alpha)\beta c_{max}}\|\nabla f(x_k)\|_2. \tag{27}$$

Since $\delta_{zero,k} \leq \kappa_4\|\nabla f(x_k)\|_2$, it follows that after at most $n_1 + 1$ reductions,

$$\delta < \frac{c_{min}^2}{4\beta c_{max}}\|\nabla f(x_k)\|_2. \tag{28}$$

and

$$\delta \leq \frac{c_{min}^2(1-\alpha)}{2(\beta+\gamma)c_{min}^2 c_{max} + 2(1-\alpha)\beta c_{max}} \|\nabla f(x_k)\|_2. \tag{29}$$

Thus, clearly

$$\delta < \frac{c_{min}^2}{2\beta c_{max}} \frac{\|\nabla f(x_k)\|_2}{2} \quad \text{and} \quad \delta \leq \frac{c_{min}^2(1-\alpha)}{(\beta+\gamma)c_{min}^2 + (1-\alpha)\beta c_{max}} \frac{\|\nabla f(x_k)\|_2}{2}. \tag{30}$$

Defining $\varepsilon = \|\nabla f(x_k)\|_2/2 < \|\nabla f(x_k)\|_2$, by Lemma 2.4, it follows that for the value of δ that fulfills (30) (so, after at most $n_1 + 1$ reductions of δ), the condition (6) takes place. By the definition of n_1 this completes the proof. □

From now on we will analyze the behavior of the algorithm with respect to second-order convergence. We will assume that the model $M(x, s)$ is a second-order approximation of the increment $f(x + s) - f(x)$ and that, at a generic iteration, the subspace S_k contains a negative-curvature direction. The first consequence will be that, if the trust region is small enough, the model decreases as the square of the trust-region radius.

The fact that the model is a higher-order approximation of the objective function will be represented by the assumption (5). Under assumptions (4) and (5), the following lemma quantifies the model decrease at a typical iteration of Algorithm 2.1. We will also see that, if d is a negative-curvature direction and δ is small enough, then s_{trial} satisfies the condition

$$M(x_k, s_{trial}) \leq \frac{\delta^2}{4} d^T \nabla^2 f(x_k) d. \tag{31}$$

Lemma 2.7 *In addition to (4), assume that (5) also holds. Let x_k be a generic iterate of Algorithm 2.1 at which the algorithm does not stop. Let s_{trial} and δ be defined as in Step 3 of the algorithm. Suppose that there exists $d \in S_k$ such that*

$$d^T \nabla^2 f(x_k) d < 0, \quad \nabla f(x_k)^T d \leq 0 \quad \textit{and} \quad \|d\|_{x_k} = 1. \tag{32}$$

Then,

$$M(x_k, s_{trial}) \leq \frac{\delta^2}{2} d^T \nabla^2 f(x_k) d + \frac{\beta_2 \delta^3}{c_{min}^3}. \tag{33}$$

Moreover, if

$$\delta \leq \frac{c_{min}^3}{4\beta_2} |d^T \nabla^2 f(x_k) d| \tag{34}$$

we have that (31) holds.

Proof. By (5), for all $s \in \mathbb{R}^n$, we have:

$$M(x_k, s) \leq \nabla f(x_k)^T s + \frac{1}{2} s^T \nabla^2 f(x_k) s + \beta_2 \|s\|_2^3.$$

Therefore, by (2) and (32), we have that

$$M(x_k, \delta d) \leq \delta \nabla f(x_k)^T d + \frac{\delta^2}{2} d^T \nabla^2 f(x_k) d + \beta_2 \delta^3 \|d\|_2^3 \leq \frac{\delta^2}{2} d^T \nabla^2 f(x_k) d + \frac{\beta_2 \delta^3}{c_{min}^3}.$$

Since s_{trial} is a minimizer of $M(x_k, s)$ onto $\|s\|_{x_k} \leq \delta$, we obtain (33). The proof that (34) implies (31) follows from a straightforward calculation. □

The following lemma shows that the sufficient descent condition holds if the trust-region radius is small enough. No assumption is made here with respect to the size of the gradient. However, we will assume the following Taylor-type property

$$f(x+s) \leq f(x) + \nabla f(x)^T s + \frac{1}{2} s^T \nabla^2 f(x) s + \gamma_2 \|s\|_2^3. \tag{35}$$

for all $x, s \in \mathbb{R}^n$.

Lemma 2.8 *Assume that (5), (3), and (35) hold. Let x_k be a generic iterate of Algorithm 2.1 at which the algorithm does not stop. Let s_{trial} and δ be defined as in Step 3 of the algorithm. Suppose that there exists $d \in S_k$ such that (32) holds. Moreover, assume that*

$$\delta \leq (1-\alpha) \frac{c_{min}^3 |d^T \nabla^2 f(x_k) d|}{4(\beta_2 + \gamma_2)}. \tag{36}$$

Then, s_{trial} satisfies (6).

Proof. By (36) we have that (34) holds. So, by Lemma 2.7, (31) also holds. By (5) and (35) we have that, for all $s \in \mathbb{R}^n$,

$$f(x_k + s) - f(x_k) \leq M(x_k, s) + (\beta_2 + \gamma_2)\|s\|_2^3. \tag{37}$$

So, by (2),

$$f(x_k + s) - f(x_k) \leq M(x_k, s) + \frac{\beta_2 + \gamma_2}{c_{min}^3} \|s\|_{x_k}^3 \tag{38}$$

for all $s \in \mathbb{R}^n$.

By (31), $M(x_k, s_{trial}) < 0$. So, by (38),

$$\frac{f(x_k + s_{trial}) - f(x_k)}{M(x_k, s_{trial})} \geq 1 + \frac{\beta_2 + \gamma_2}{c_{min}^3 M(x_k, s_{trial})} \|s_{trial}\|_{x_k}^3. \tag{39}$$

Thus, as $\|s_{trial}\|_{x_k} \leq \delta$,

$$\frac{f(x_k + s_{trial}) - f(x_k)}{M(x_k, s_{trial})} \geq 1 + \frac{\beta_2 + \gamma_2}{c_{min}^3 M(x_k, s_{trial})} \delta^3. \tag{40}$$

Then, by (31),

$$\frac{f(x_k + s_{trial}) - f(x_k)}{M(x_k, s_{trial})} \geq 1 + \frac{4(\beta_2 + \gamma_2)}{c_{min}^3 d^T \nabla^2 f(x_k) d} \delta. \tag{41}$$

Now, by (36),

$$\frac{4(\beta_2 + \gamma_2)}{c_{min}^3 |d^T \nabla^2 f(x_k) d|} \delta \leq 1 - \alpha.$$

Then, since $d^T \nabla^2 f(x_k) d < 0$,

$$\frac{4(\beta_2 + \gamma_2)}{c_{min}^3 d^T \nabla^2 f(x_k) d} \delta \geq \alpha - 1.$$

Therefore, by (41),

$$\frac{f(x_k + s_{trial}) - f(x)}{M(x_k, s_{trial})} \geq \alpha, \tag{42}$$

which implies that (6) holds. □

In the following lemma we prove that, under the assumption of existence of a sufficiently negative curvature direction in S_k, the accepted trust-region radius δ_k is bounded below by a positive quantity, the model decreases sufficiently (see (45)), and the objective function value decreases more than a well defined quantity (see (46)).

Lemma 2.9 *Assume that for all $x, s \in \mathbb{R}^n$, (3), (35), (4), and (5) hold. Let $\epsilon_2 > 0$ be arbitrary. Let $K \subseteq \mathbb{N}$ be the set of indices for which there exists $v_k \in S_k$ such that*

$$v_k^T \nabla^2 f(x_k) v_k \leq -\epsilon_2 \text{ and } \|v_k\|_2 = 1. \tag{43}$$

Then, for all $k \in K$ we have that the accepted trust-region radius δ_k satisfies:

$$\delta_k \geq \min\left\{\delta_{zero,k}, \eta\epsilon_2(1-\alpha)\frac{c_{min}^3}{4c_{max}^2(\beta_2+\gamma_2)}\right\}. \tag{44}$$

Moreover, for all $k \in K$,

$$M(x_k, s_k) \leq -\kappa_5 \min\{\delta_{zero,k}^2\epsilon_2, \epsilon_2^3\} \tag{45}$$

and

$$f(x_{k+1}) \leq f(x_k) - \alpha\kappa_5 \min\{\delta_{zero,k}^2\epsilon_2, \epsilon_2^3\}, \tag{46}$$

where

$$\kappa_5 = \min\left\{\frac{1}{4c_{max}^2}, \frac{\eta^2\kappa_6^2}{4c_{max}^2}, \frac{\kappa_6^2}{4c_{max}^2}\right\} \tag{47}$$

and

$$\kappa_6 = (1-\alpha)\frac{c_{min}^3}{4c_{max}^2(\beta_2+\gamma_2)}. \tag{48}$$

Proof. By (43), for all $k \in K$, we have that $\|v_k\|_2 = 1$ and :

$$\frac{v_k^T}{\|v_k\|_{x_k}} \nabla^2 f(x_k) \frac{v_k}{\|v_k\|_{x_k}} \leq -\frac{\epsilon_2}{\|v_k\|_{x_k}^2}, \text{ with } \left\|\frac{v_k}{\|v_k\|_{x_k}}\right\|_{x_k} = 1. \tag{49}$$

Define $d_k = \frac{v_k}{\|v_k\|_{x_k}}$ if $\nabla f(x_k)^T v_k \leq 0$ and $d_k = -\frac{v_k}{\|v_k\|_{x_k}}$ if $\nabla f(x_k)^T v_k > 0$. Then, by (2), (43), and (49),

$$d_k^T \nabla^2 f(x_k) d_k \leq -\frac{\epsilon_2}{c_{max}^2}, \nabla f(x_k)^T d_k \leq 0, \text{ and } \|d_k\|_{x_k} = 1. \tag{50}$$

By Lemma 2.8, if

$$\delta \leq |d_k^T \nabla^2 f(x) d_k|(1-\alpha)\frac{c_{min}^3}{4(\beta_2+\gamma_2)}, \tag{51}$$

the sufficient condition (6) is satisfied and, by Lemma 2.7,

$$M(x_k, s_{trial}) \leq \frac{\delta^2}{4} d_k^T \nabla^2 f(x_k) d_k. \tag{52}$$

Therefore, by (50), we have that (6) and (52) hold whenever

$$\delta \leq \frac{\epsilon_2}{c_{max}^2}(1-\alpha)\frac{c_{min}^3}{4(\beta_2+\gamma_2)}. \tag{53}$$

If the first choice of $\delta = \delta_{zero,k}$ does not satisfy (6), after a finite number of reductions of δ we obtain (53). Thus, after a finite number of reductions of δ we obtain (6). Let us denote by $\delta_{previous}$ the trust-region radius tested before the accepted δ_k. Clearly, for the accepted δ that satisfies (6), we have that $\delta_k \geq \eta\delta_{previous}$ and, since $\delta_{previous}$ does not satisfy (6), it does not satisfy (53) either. So, by (48),

$$\delta_{previous} > \kappa_6\epsilon_2.$$

Therefore,

$$\delta_k > \eta\kappa_6\epsilon_2.$$

So, by (48), (44) is proved.

We proved above that (53) implies (52). Therefore, if

$$\delta_k \leq \kappa_6\epsilon_2 \tag{54}$$

we have that, by (50),

$$M(x_k, s_k) \leq -\frac{\delta_k^2\epsilon_2}{4c_{max}^2}. \tag{55}$$

Consider now the case in which (54) does not hold. So, $\delta_k > \kappa_6\epsilon_2$. Define $\delta = \kappa_6\epsilon_2$. So, $\delta < \delta_k$. Again, as (53) implies (52) and s_{trial} is the minimizer of M corresponding to δ, we have that $M(x_k, s_{trial}) \leq -\frac{\delta^2\epsilon_2}{4c_{max}^2}$. But, since $\|s_{trial}\|_{x_k} \leq \delta < \delta_k$ and s_k is the minimizer of the model on the trust region defined by δ_k, we deduce that

$$M(x_k, s_k) \leq M(x_k, s_{trial}) \leq -\frac{\delta^2\epsilon_2}{4c_{max}^2}.$$

Hence, since $\delta = \kappa_6\epsilon_2$,

$$M(x_k, s_k) \leq -\frac{\kappa_6^2\epsilon_2^3}{4c_{max}^2}. \tag{56}$$

Therefore, combining (55) and (56),

$$M(x_k, s_k) \leq \max\left\{ -\frac{\delta_k^2\epsilon_2}{4c_{max}^2}, -\frac{\kappa_6^2\epsilon_2^3}{4c_{max}^2} \right\}.$$

Thus, by (44), when $k \in K$ we have:

$$M(x_k, s_k) \leq \max\left\{ -\frac{\min\{\delta_{zero,k}, \eta\kappa_6\epsilon_2\}^2\epsilon_2}{4c_{max}^2}, -\frac{\kappa_6^2\epsilon_2^3}{4c_{max}^2} \right\}. \tag{57}$$

So,

$$\begin{aligned} -M(x_k, s_k) &\geq -\max\left\{ -\frac{\min\{\delta_{zero,k}, \eta\kappa_6\epsilon_2\}^2\epsilon_2}{4c_{max}^2}, -\frac{\kappa_6^2\epsilon_2^3}{4c_{max}^2} \right\} \\ &= \min\left\{ \frac{\min\{\delta_{zero,k}, \eta\kappa_6\epsilon_2\}^2\epsilon_2}{4c_{max}^2}, \frac{\kappa_6^2\epsilon_2^3}{4c_{max}^2} \right\} \\ &= \min\left\{ \frac{\min\{\delta_{zero,k}^2, \eta^2\kappa_6^2\epsilon_2^2\}\epsilon_2}{4c_{max}^2}, \frac{\kappa_6^2\epsilon_2^3}{4c_{max}^2} \right\} \end{aligned}$$

$$= \min\left\{\frac{\delta_{zero,k}^2\epsilon_2}{4c_{max}^2}, \frac{\eta^2\kappa_6^2\epsilon_2^3}{4c_{max}^2}, \frac{\kappa_6^2\epsilon_2^3}{4c_{max}^2}\right\}$$

$$\geq \min\left\{\frac{1}{4c_{max}^2}, \frac{\eta^2\kappa_6^2}{4c_{max}^2}, \frac{\kappa_6^2}{4c_{max}^2}\right\} \min\{\delta_{zero,k}^2\epsilon_2, \epsilon_2^3\}.$$

Therefore, by (6), (47) and (48), (46) is also proved, and the result is established. □

The following theorem gives an upper bound on the number of iterations at which a sufficiently negative curvature direction in the subspace S_k may exist. This is a complexity result which, essentially, says that the existence of such direction is possible at most in $O(\epsilon_2^{-3})$ iterations. A complementary result establishes an upper bound on the number of function evaluations at such iterations.

Theorem 2.3 *Assume that for all $x, s \in \mathbb{R}^n$, (3), (35), (4), and (5) hold. Let f_{min} be a lower bound of f onto $\mathbb{R}^n$. Let $\epsilon_2 > 0$ be arbitrary. Let $K \subseteq \mathbb{N}$ be the set of indices for which there exists $v_k \in S_k$ such that (43) holds. Denote by $\#K$ the number of indices in K. Then:*

1. *If there exists $\delta_{\min} > 0$ such that*

$$\delta_{zero,k} \geq \delta_{\min} \text{ for all } k \in K, \tag{58}$$

we have that:

$$\#K \leq \frac{f(x_0) - f_{min}}{\alpha\kappa_5 \min\{\delta_{\min}^2\epsilon_2, \epsilon_2^3\}}, \tag{59}$$

where κ_5 is defined by (47).

2. *If there exists $\kappa_7 > 0$ such that*

$$\delta_{zero,k} \geq \kappa_7\epsilon_2 \text{ for all } k \in K, \tag{60}$$

we have that

$$\#K \leq \frac{f(x_0) - f_{min}}{\alpha\kappa_8\epsilon_2^3}, \tag{61}$$

where $\kappa_8 = \min\{\kappa_5\kappa_7, \kappa_5\}$ and κ_5 is given by (47).

3. *The number of (function and derivatives) evaluations performed by the algorithm at iteration k is bounded above by $\lfloor \log(\epsilon_2)/\log(1-\eta) + \kappa_9 \rfloor$ in the case of (58), and by $\lfloor \log(\kappa_7\epsilon_2/\delta_{\max})/\log(1-\eta) + \kappa_9 \rfloor$ in the case of (60), where*

$$\kappa_9 = \log\left[\frac{\eta}{\delta_{\max}c_{max}^2}(1-\alpha)\frac{c_{min}^3}{4(\beta_2+\gamma_2)}\right] / \log(1-\eta) + \log(\delta_{\min}/\delta_{\max})/\log(1-\eta)$$

in the case of (58) and

$$\kappa_9 = \log\left[\frac{\eta}{\delta_{\max}c_{max}^2}(1-\alpha)\frac{c_{min}^3}{4(\beta_2+\gamma_2)}\right] / \log(1-\eta) + \log(\kappa_7\epsilon_2/\delta_{\max})/\log(1-\eta)$$

in the case of (60).

Proof. In order to prove the first part of the thesis, note that, since $f(x_{k+1}) \leq f(x_k)$ for all $k \in \mathbb{N}$, then, by Lemma 2.9, the number of iterations at which (46) holds cannot exceed

$$\left\lfloor \frac{f(x_0) - f_{min}}{\alpha\kappa_5 \min\{\delta_{\min}^2\epsilon_2, \epsilon_2^3\}} \right\rfloor.$$

So, (59) is proved. The proof of (61) follows as in (59) replacing $\delta_{\min}$ with $\kappa_7\epsilon_2$.

Now, by (44), for all $k \in K$, a sufficient condition for the acceptance of a trust-region radius is:

$$\delta \leq \min\left\{\delta_{zero,k}, \eta\frac{\epsilon_2}{c_{max}^2}(1-\alpha)\frac{c_{min}^3}{4(\beta_2+\gamma_2)}\right\}. \tag{62}$$

If the first trust-region radius $\delta_{zero,k}$ is accepted, the third part of the thesis is obviously true. Otherwise, let us analyze what happens if it is reduced n_{red} times. After this number of reductions the radius will be between $\eta^{n_{red}}\delta_{zero,k}$ and $(1-\eta)^{n_{red}}\delta_{zero,k}$. Since $\delta_{zero,k} \leq \delta_{\max}$, after n_{red} reductions it will be smaller than $(1-\eta)^{n_{red}}\delta_{\max}$. Then, a sufficient condition for the acceptance of the trust-region radius after n_{red} reductions is:

$$(1-\eta)^{n_{red}}\delta_{\max} \leq \min\left\{\delta_{zero,k}, \eta\frac{\epsilon_2}{c_{max}^2}(1-\alpha)\frac{c_{min}^3}{4(\beta_2+\gamma_2)}\right\}. \tag{63}$$

In the case of (58), as $\delta_{zero,k} \geq \delta_{\min}$, a sufficient condition for the fulfillment of (63) is:

$$(1-\eta)^{n_{red}}\delta_{\max} \leq \min\left\{\delta_{\min}, \eta\frac{\epsilon_2}{c_{max}^2}(1-\alpha)\frac{c_{min}^3}{4(\beta_2+\gamma_2)}\right\}. \tag{64}$$

Now, (64) holds if the following two conditions hold:

$$(1-\eta)^{n_{red}}\delta_{\max} \leq \delta_{\min} \tag{65}$$

and

$$(1-\eta)^{n_{red}}\delta_{\max} \leq \eta\frac{\epsilon_2}{c_{max}^2}(1-\alpha)\frac{c_{min}^3}{4(\beta_2+\gamma_2)}. \tag{66}$$

Thus, the acceptance condition (64) holds if the following conditions hold:

$$n_{red}\log(1-\eta) \leq \log(\delta_{\min}/\delta_{\max}) \tag{67}$$

and

$$n_{red}\log(1-\eta) \leq \log\left[\epsilon_2\frac{\eta}{\delta_{\max}c_{max}^2}(1-\alpha)\frac{c_{min}^3}{4(\beta_2+\gamma_2)}\right]. \tag{68}$$

Therefore, the trust-region radius will be accepted whenever

$$n_{red} \geq \log(\delta_{\min}/\delta_{\max})/\log(1-\eta) \tag{69}$$

and

$$n_{red} \geq \log\epsilon_2/\log(1-\eta) + \log\left[\frac{\eta}{\delta_{\max}c_{max}^2}(1-\alpha)\frac{c_{min}^3}{4(\beta_2+\gamma_2)}\right]/\log(1-\eta). \tag{70}$$

In the case of (60) the same reasoning leads to the proof that the trust-region radius will be accepted whenever (70) holds and

$$n_{red} \geq \log(\kappa_7\epsilon_2/\delta_{\max})/\log(1-\eta). \tag{71}$$

This completes the proof of the theorem. □

We finish this section deducing an asymptotic corollary.

Corollary 2.1 *Let us assume the hypotheses of Theorem 2.3. Let us also assume that we stop the algorithm when*

$$\nabla f(x_k) = 0 \text{ and } v^T \nabla^2 f(x_k) v \geq 0 \text{ for all } v \in S_k.$$

Then, either the algorithm stops in a finite number of iterations or

$$\lim_{k \to \infty} \nabla f(x_k) = 0 \tag{72}$$

and, for every choice of $v_k \in S_k$ such that $\|v_k\|_2 = 1$,

$$\liminf v_k^T \nabla^2 f(x_k) v_k \geq 0. \tag{73}$$

Moreover, if x_* is a limit point of $\{x_k\}$ and v is a limit point of $\{v_k\}$, where $v_k \in S_k$ and $\|v_k\|_2 = 1$ for all $k \in K$, we have that

$$\nabla f(x_*) = 0 \tag{74}$$

and

$$v^T \nabla^2 f(x_*) v \geq 0. \tag{75}$$

Proof. In Theorem 2.1 we already proved (72) and (74). Now, if (73) is not true, there exist an infinite set of indices K_1 and some $\epsilon > 0$ such that for all $k \in K_1$ there exists $v_k \in S_k$, $\|v_k\|_2 = 1$, that satisfies

$$v_k^T \nabla^2 f(x_k) v_k \leq -\epsilon.$$

This is impossible because, according to Theorem 2.3, K_1 must be finite. Now, if (75) does not hold we have that $v^T \nabla^2 f(x_*) v < 0$ and, by continuity, $v_k^T \nabla^2 f(x_k) v_k < \frac{v^T \nabla^2 f(x_*) v}{2} < 0$ for $k \in K$ large enough. This contradicts (73). □

3 Separable cubic modeling and Lanczos method

The efficiency of Algorithm 2.1 is linked to the choice of the model $M(x_k, s)$ and the subspace S_k. Following the developments in [24, 25], let us start by considering the following third-order secant perturbation of the second-order Taylor approximation of $f(x_k + s) - f(x_k)$:

$$M_k(s) = \nabla f(x_k)^T s + \frac{1}{2} s^T H_k s + \frac{1}{6} \sum_{i=1}^{n} \rho_k^i s_i^3, \tag{76}$$

where $H_k = \nabla^2 f(x_k)$ is the Hessian of f at x_k, and the vector $\rho_k \in \mathbb{R}^n$ of parameters ρ_k^i for $1 \leq i \leq n$ are chosen, based on the secant equation, as described in [24]. Notice that (76) may be non-convex and can have local non-global minima. Notice also that the cubic model (76) satisfies (4) and (5).

Concerning **Step 2** of Algorithm 2.1, instead of minimizing the cubic model (76) in the whole space $\mathbb{R}^n$, as proposed in the scheme developed in [24], for which a Schur factorization of H_k is required, we will consider the minimization of (76) subject to $s \in S_k$, where S_k is a

properly chosen low-dimensional subspace of $\mathbb{R}^n$. Since H_k could be indefinite, then S_k must be generated by a basis that includes directions of negative curvature, as well as directions of positive curvature; which is a key issue to avoid the convergence of the optimization scheme to local-nonglobal minimizers. Since we can only choose a few vectors, say $p \ll n$, it is highly convenient to include approximate eigenvectors of H_k associated with its extreme eigenvalues, i.e., the smallest and the largest ones. The suitable matrix-free mechanism to build a basis $\{v_1, v_2, \ldots, v_p\}$ of S_k, such that this task is accomplished in a natural way, is the Lanczos method [20, 21].

3.1 Lanczos method for building a subspace

Lanczos method is a commonly used method for large symmetric eigenvalue problems and sparse SVD problems; see, e.g., [13, 31, 32]. However, it has also been used for nonconvex large-scale optimization, specially to compute good directions of negative curvature when solving the quadratic model trust-region subproblems; see, e.g., [9, 14, 15, 16, 27, 30].

In our setting, given the symmetric matrix $H_k \in \mathbb{R}^{n\times n}$, a small positive integer p, and an initial unit vector v_1, at every iteration k of Algorithm 2.1 we will apply Lanczos method to build an $n \times p$ matrix V_k with orthonormal columns v_i for $1 \leq i \leq p$, and a tridiagonal $p \times p$ matrix T_k, such that

$$T_k = V_k^T H_k V_k. \tag{77}$$

An advantage of Lanczos method is that it can be started with an arbitrary initial unit vector v_1 without altering the structure of the method. As discussed in Section 2, to guarantee convergence to first-order stationary points we need that $\nabla f(x_k) \in S_k$, and hence a natural and suitable choice in our setting is $v_1 = \nabla f(x_k)/\|\nabla f(x_k)\|_2$. For the sake of completeness we now present our special version of the Lanczos algorithm:

Lanczos Algorithm

Given the symmetric matrix H_k, a positive integer p and the vector $\nabla f(x_k)$,
set $v_1 = \nabla f(x_k)/\|\nabla f(x_k)\|_2$, $\beta_0 = 0$ and $v_0 = 0$.

for $j = 1, 2, \ldots, p$
  compute $w = H_k v_j - \beta_{j-1} v_{j-1}$, $\alpha_j = v_j^T w$,
    $w = w - \alpha_j v_j$ and $\beta_j = \|w\|_2$.
  if $\beta_j = 0$, **update** $p \leftarrow j$ and **stop**.
  else $v_{j+1} = w/\beta_j$
  end if
end for

If β_j is nonzero for $1 \leq j \leq p$ (i.e., if no breakdown occurs), then Lanczos Algorithm generates the $n \times p$ matrix V_k such that $V_k^T V_k = I$, and the symmetric tridiagonal matrix T_k with α_i, $1 \leq i \leq p$, in the main diagonal, and β_i, $1 \leq i \leq p-1$, in the main sub-diagonals.

Notice that Lanczos method is matrix-free since it does not require forming or storing the Hessian H_k. In particular, only the action of the matrix H_k on a given vector is required and

these matrix-vector products can be obtained by means of a specialized routine, or they can be approximated using finite-difference formulas; see [4].

Notice also that if $p = n$, then V_k is an orthogonal matrix and so (77) establishes a similarity transformation. In that case, T_k and H_k have the same eigenvalues. If $p < n$ then the eigenvalues of T_k (called Ritz values) approximate some of the eigenvalues of H_k. In particular, when $p \geq 1$ increases, the extreme eigenvalues of T_k, i.e., the smallest and the largest ones, converge first, and the interior eigenvalues converge last; which is exactly what we need in our setting for small values of p. Moreover, convergence is monotonic with the i-th largest (smallest) eigenvalue of T_k increasing (decreasing) to the i-th largest (smallest) eigenvalue of H_k, as long as the Lanczos Algorithm does not stop prematurely with some $\beta_j = 0$ for $j < p$. For details see [13, Ch. 7].

Using (77), instead of minimizing the cubic model (76) subject to $s \in S_k$, we consider vectors of the form $s = V_k z$, and find $s_k = V_k z_k$ where $z_k \in \mathbb{R}^p$ minimizes the following cubic model

$$\widetilde{M}_k(z) = (V_k^T \nabla f(x_k))^T z + \frac{1}{2} z^T T_k z + \frac{1}{6} \sum_{i=1}^{p} \tilde{\rho}_k^i z_i^3, \tag{78}$$

where $\tilde{\rho}_k = V_k^T \rho_k$ is a vector in $\mathbb{R}^p$. If a breakdown occurs in the Lanczos process, i.e. if $\beta_j = 0$ for some $j < p$ (conveniently referred as a lucky breakdown), then immediately an invariant subspace of the matrix H_k has been obtained. Although this happens seldom in practice, in that case the minimizer of the quadratic model (first two terms in (76)) coincides with the minimizer of the reduced quadratic model (first two terms in (78)) for which V_k only has $j < p$ orthonormal columns; see [9].

3.2 Separable cubic modeling

In order to minimize independently p-cubic one-dimensional functions, let us now consider the Schur factorization of the tridiagonal matrix T_k, that was previously obtained using Lanczos method:

$$T_k = Q_k D_k Q_k^T, \tag{79}$$

where Q_k is an orthogonal $p \times p$ matrix whose columns are the eigenvectors of T_k, and D_k is a real diagonal $p \times p$ matrix whose diagonal entries are the eigenvalues of T_k. Notice that since T_k is symmetric then (79) is well-defined for all k. Moreover, since $p \ll n$, the cost of computing (79) is not significant. The columns of the matrix $V_k Q_k$ are known as Ritz vectors. When $p \geq 1$ increases, the Ritz vectors approximate the eigenvectors of H_k associated with the corresponding Ritz values, i.e., associated with the diagonal elements of D_k; see [13, 31, 32].

Let us define $\widehat{Q}_k = V_k Q_k$, and let us consider the following change of variables:

$$y = Q_k^T z = Q_k^T V_k^T s = (V_k Q_k)^T s = \widehat{Q}_k^T s.$$

Inspired by the cubic model (78), our proposal is to use, at each iteration, the separable cubic model

$$\widehat{M}_k(y) = (\widehat{Q}_k^T \nabla f(x_k))^T y + \frac{1}{2} y^T D_k y + \frac{1}{6} \sum_{i=1}^{p} \widehat{\rho}_k^i y_i^3, \tag{80}$$

where now $\widehat{\rho}_k = Q_k^T \tilde{\rho}_k = \widehat{Q}_k^T \rho_k$ is a vector in $\mathbb{R}^p$. Therefore, the subproblem

$$\min_{y \in \mathbb{R}^p} \widehat{M}_k(y)$$

must be solved at every k to compute the vector y_k, and then we recover a step in $\mathbb{R}^n$

$$s_k = \widehat{Q}_k y_k.$$

Notice that the gradient of the model $\widehat{M}_k(y)$ in (80) is given by

$$\nabla \widehat{M}_k(y) = \widehat{Q}_k^T \nabla f(x_k) + D_k y + \frac{1}{2}\widehat{w}_k,$$

where $\widehat{w}_k \in \mathbb{R}^p$ and its i-th entry is equal to $\widehat{\rho}_k^i y_i^2$. Similarly, the Hessian of (80) is given by

$$\nabla^2 \widehat{M}_k(y) = D_k + \mathrm{diag}(\widehat{\rho}_k^i y_i).$$

For choosing the parameters $\widehat{\rho}_k^i$, $1 \le i \le p$, in the cubic model (80), we follow the proposal in [24] which is inspired by the classical secant equation:

$$\widehat{\rho}_k^i = \frac{(D_k - \widehat{Q}_k^T H_{k-1} \widehat{Q}_k)_{ii}}{(\widehat{Q}_k^T s_{k-1})_i}, \tag{81}$$

which is well-defined as long as $(\widehat{Q}_k^T s_{k-1})_i \neq 0$ for all i. Since $(\widehat{Q}_k^T s_{k-1})_i$ could be very close to zero for some i, we impose a safeguard at (81) before computing $\widehat{\rho}_k^i$: given a small $\epsilon > 0$, if $-\epsilon < (\widehat{Q}_k^T s_{k-1})_i < 0$, we set $(\widehat{Q}_k^T s_{k-1})_i = -\epsilon$; and if $0 < (\widehat{Q}_k^T s_{k-1})_i < \epsilon$, we set $(\widehat{Q}_k^T s_{k-1})_i = \epsilon$. In a practical implementation it suffices to choose $\epsilon = \sqrt{\mu}$ where μ is the unit roundoff.

3.3 A trust-region strategy

To guarantee convergence to local minimizers of (1), the separable cubic model (80) can be embedded in a Trust-Region (TR) globalization strategy, that in our case will be a special case of the model Algorithm 2.1. In particular, for a given $\delta_k > 0$, the minimization of $\widehat{M}_k(y)$ subject to $s \in S_k$ and $\|s\|_{x_k} \le \delta_k$ is reduced to a p-dimensional trust-region subproblem, of the type analyzed in [24], using now the separable cubic model in (80).

For TR globalization strategies, the step length δ_k is chosen previously to the computation of the step s_k. In order to update the size of the trust region, once iteration k is completed, the values of the so-called actual reduction (Ared) and predicted reduction (Pred) are computed as follows:

$$\text{Ared } = f(x_k) - f(x_k + s_k) \text{ and } \text{Pred } = M_k(0) - M_k(s_k) = -M_k(s_k).$$

The quotient of these two quantities, $R = \text{Ared}/\text{Pred}$, is then used to decide if δ_k is increased, decreased, or left the same for the next iteration; see e.g., [9] for details in the quadratic modeling case.

The cost of solving the subproblems in our specialized TR strategy reduces to the bound minimization of p independent one-dimensional cubic real valued polynomials. To be precise, the subproblem that must be solved at iteration k is the following:

$$\min \widehat{M}_k(y) \quad \text{subject to} \quad -\delta_k \le y_i \le +\delta_k, \quad \text{for all } \ 1 \le i \le p. \tag{82}$$

For the sake of completeness, we now briefly discuss how to find the global minimizer of a general cubic polynomial in one real variable, say

$$P_3(z) = c_0 + c_1 z + c_2 z^2 + c_3 z^3,$$

on a closed and bounded interval $[-\delta, +\delta]$ for $\delta > 0$; for additional details see [24, 25]. Let us assume that $c_3 \neq 0$. To compute the critical points of $P_3(z)$, we solve the quadratic equation $P_3'(z) = c_1 + 2c_2 z + 3c_3 z^2 = 0$. If the discriminant of $P_3'(z)$ is negative, i.e. if $\Delta = 4c_2^2 - 12c_3c_1 < 0$, then $P_3(z)$ has no real local minimum or maximum, and hence the bounded global minimizer is given by

$$z^* = \text{ argmin } \{P_3(-\delta), P_3(+\delta)\}. \tag{83}$$

If $\Delta \geq 0$, we compute the critical points $(-2c_2 \pm \sqrt{\Delta})/(6c_3)$ and choose the one that yields the minimum value at $P_3(z)$: z_{lmin}. If $z_{lmin} \in (-\delta, +\delta)$ then the bounded global minimizer is given by

$$z^* = \text{ argmin } \{P_3(-\delta), P_3(z_{lmin}), P_3(+\delta)\}. \tag{84}$$

If either $z_{lmin} < -\delta$ or $z_{lmin} > +\delta$ then the bounded global minimizer is given by (83).

For the sake of clarity and completeness, we now present the specialized TR algorithm for the separable cubic model (80), which shares some similarities with Algorithm 5.1 in [24], but that also has several fundamental differences that make it suitable for large-scale problems, mainly that it is applied on the reduced p-dimensional subspace S_k, using Lanczos method.

Algorithm 3.1 (Specialized TR algorithm for the separable cubic model (80))

Given $x_0 \in \mathbb{R}^n$, $\nabla f(x_0)$, $H_0 = H_0^T$, a small integer $1 \leq p < n$, a large integer $kmax > 1$, $\delta_{\max} \geq \delta_0 \geq \delta_{\min} > 0$, $\widehat{\rho}_0 \in \mathbb{R}^p$, $0 < \eta_s < \eta_v < 1$, $\gamma > 1$, $0 < \gamma_d < 1$, $tol > tol_1 > 0$, and $\tilde{\rho} > 0$; use Lanczos Algorithm with p, $\nabla f(x_0)$ and H_0 to generate V_0 and T_0; use then Schur factorization with T_0 to generate Q_0 and D_0; and set $\widehat{Q}_0 = V_0 Q_0$. Set $k = 0$.

while ($\|\nabla f(x_k)\|_2 > tol$ and $k \leq kmax$) **do**

Step 1: **set** $\delta_k = \min\{\max\{\delta_k, \delta_{\min}\}, \delta_{\max}\}$ and **compute** $b_k = \widehat{Q}_k^T \nabla f(x_k)$.

Step 2: **solve** the TR subproblem (82) **for** y_k:

set $\delta = \delta_k$, $c_0 = 0$, $c_1 = (b_k)_i$, $c_2 = \frac{1}{2}(D_k)_{ii}$, and $c_3 = \frac{1}{6}\widehat{\rho}_k^i$.

for $i = 1, 2, \ldots, p$

set $(y_k)_i = z^*$ using (83) or (84).

end for

Step 3: **set** $s_k = \widehat{Q}_k y_k$, Ared $= f(x_k) - f(x_k + s_k)$, and Pred $= -M_k(s_k)$.

if |Pred| $< tol_1$, **stop** and **report** x_k.

else set $R =$ Ared/Pred.

if $R \geq \eta_v$, **set** $x_{k+1} = x_k + s_k$ and $\delta_{k+1} = \gamma \delta_k$.

else if $R \geq \eta_s$, **set** $x_{k+1} = x_k + s_k$ and $\delta_{k+1} = \delta_k$.

else $\delta_k = \gamma_d \delta_k$ and go to **Step 2.**

end if

end if

end if

Step 4: **evaluate** $H_{k+1} = H_{k+1}^T$ and $\nabla f(x_{k+1})$ at x_{k+1}. **Use** Lanczos Algorithm with p, $\nabla f(x_{k+1})$ and H_{k+1} to **generate** V_{k+1} and T_{k+1}; **use** then Schur factorization with T_{k+1} to **generate** Q_{k+1} and D_{k+1}; and **set** $\widehat{Q}_{k+1} = V_{k+1}Q_{k+1}$.

Step 5: **set** $k = k + 1$, **compute** $\widehat{\rho}_k \in \mathbb{R}^p$ using (81), and **set** $\widehat{\rho}_k^i = \min\{\max\{\widehat{\rho}_k^i, -\tilde{\rho}\}, \tilde{\rho}\}$ for $1 \leq i \leq p$.

end while

Notice that Algorithm 3.1 can be viewed as a special case of Algorithm 2.1, when the involved model $M(x, s)$ is given by (80). We stop the process when the norm of the gradient is sufficiently small, in which case we obtain an approximate solution. We also stop the process when a maximum number of iterations ($kmax$) has been reached. In addition, at Step 3, if |Pred| is very close to zero we avoid the computation of R and stop the process reporting x_k as the approximate solution. In that case, the size of s_k is numerically zero and there is no possible further improvement to minimize the function f. Otherwise, the step s_k is accepted and the new iterate x_{k+1} is set to $x_k + s_k$ whenever the quotient R is greater than η_s. If in addition R is greater than $\eta_v > \eta_s$ then the size of the trust region δ_k is increased. If R is smaller than η_s, which means that the agreement between the cubic model and the function is insufficient, then the size of the trust region is decreased and the TR subproblem is solved once again with the new reduced δ_k. To satisfy the assumptions required by the theoretical results in Section 2, at the beginning of every iteration we make sure that $0 < \delta_{\min} \leq \delta_k \leq \delta_{\max}$. In practical implementations, the typical values of the key parameters are $\widehat{\rho}_0^i = 1$ for all i, $\delta_{\min} = 0.05$, $\delta_{\max} = 1. \times 10^5$, $\eta_v = 0.9$, $\eta_s = 0.01$, $\gamma = 2$, $\gamma_d = 0.5$, and $\tilde{\rho} = 1. \times 10^2$.

4 Numerical experiments

We test our algorithm on some known test functions, and also on the well-known problem of packing circles into a box. For the test functions we present some preliminary comparisons with other available strategies for large-scale optimization. Moreover, for the packing problem we also apply the global optimization solver GAMS-BARON [33] (version 12.3.3) (with default parameters settings) to compare with our results. All experiments have been performed on a CPU Xeon E5-2650 with 2 Tb of HD and 160Gb of Ram memory, using the operating system Linux. The algorithm was coded in Fortran 77 with double precision and compiled with the GNU compiler, version 6.1.0. Furthermore we used routines of the Lapack Library [22] for the Schur factorization required by the algorithm. The running times are always given in CPU seconds.

The first set of test problems considered the multi-dimensional separable function

$$\sum_{i=1}^{n} \left(\frac{1}{2} i {x_i}^2 - 5i \, \sin(x_i) \right). \tag{85}$$

As described in [24], each of the one-dimensional functions $\frac{1}{2} i {x_i}^2 - 5i \, \sin(x_i)$, $i = 1, \ldots, n$ has a local minimizer at $l \approx -3.8374$ and a global minimizer at $\tau = 1.30644$, which gives many local minimizers in $\mathbb{R}^n$ corresponding to all possible combinations of the values l and τ in the i-th entries of the vector, $i = 1, \ldots, n$. However, there is only one global minimizer x_τ with

$(x_\tau)_i = 1.30644$ for all $i = 1, \ldots, n$. We will also denote as x_l the worst possible local minimizer, i.e., such that $(x_l)_i = l$ for all $i = 1, \ldots, n$. We ran the algorithm for several dimensions of the problem and the subspace $\mathbb{R}^p$, considering the following initial guesses: $x_0^1 = (-1, -1, \ldots, -1)^T$, x_0^2 such that each component is a random number with uniform distribution in $[-1, 1]$, and $x_0^3 = (1, 1, \ldots, 1)^T$. The values of the stopping tolerances tol and tol_1 have been set to 10^{-6} and 10^{-10}, respectively. The maximum number of iterations has been set to $kmax = 3000$. The obtained results are reported in Table 1, where x^* and f^* represent the limit point at which the sequence converges and its objective value, respectively, IT and TIME are the number of required iterations and the CPU time (in seconds) to obtain such a point, and $x_{\tau\setminus\kappa}$ denotes a local minimizer that differs from x_τ in κ components. The obtained results show that when κ is a small positive integer, as compared to n, the difference between $f(x_{\tau\setminus\kappa})$ and $f(x_\tau)$ is not significant. We also note that the algorithm often finds the global minimizer of the function, or a local one for which most of its entries have the global minimizer value, even when a small subspace $\mathbb{R}^p$ is used. Moreover, in general, the increase in the dimension of the problem does not seem to affect the performance of the algorithm. In fact, except for $n = 2000$ and $p = 200$, the number of required iterations is very small.

Table 1: Performance of Algorithm 3.1 when applied to (85).

		x_0^1				x_0^2				x_0^3			
n	p	x^*	IT	f^*	TIME	x^*	IT	f^*	TIME	x^*	IT	f^*	TIME
400	5	$x_{\tau\setminus 1}$	83	-3.15E+5	1.6E-1	$x_{\tau\setminus 1}$	68	-3.15E+5	1.0E-01	$x_{\tau\setminus 1}$	60	-3.15E+5	8.9E-2
	20	x_τ	16	-3.18E+5	1.6E-1	x_τ	14	-3.18E+05	6.3E-2	x_τ	8	-3.18E+5	4.3E-2
	100	x_τ	8	-3.18E+5	5.2E-1	x_τ	8	-3.18E+05	3.8E-1	x_τ	5	-3.18E+5	2.5E-1
1000	50	$x_{\tau\setminus 3}$	12	-1.96E+6	7.4E-1	$x_{\tau\setminus 1}$	13	-1.98E+6	6.8E-1	x_τ	6	-1.98E+6	2.7E-1
	100	$x_{\tau\setminus 5}$	11	-1.95E+6	1.2E+0	x_τ	10	-1.98E+6	1.2E+0	x_τ	4	-1.98E+6	5.7E-1
	200	$x_{\tau\setminus 9}$	13	-1.91E+6	7.2E+0	$x_{\tau\setminus 2}$	10	-1.97E+6	7.9E+0	x_τ	5	-1.98E+6	2.8E+0
2000	50	$x_{\tau\setminus 4}$	16	-7.89E+6	5.5E+0	x_τ	17	-7.95E+6	4.3E+0	x_τ	8	-7.95E+6	2.4E+0
	200	$x_{\tau\setminus 147}$	3000	-7.81E+4	1.1E+3	$x_{\tau\setminus 1}$	11	-7.93E+6	1.6E+1	x_τ	5	-7.95E+6	5.7E-1
	400	$x_{\tau\setminus 35}$	34	-7.42E+6	1.9E+2	$x_{\tau\setminus 6}$	23	-7.88E+6	1.3E+2	$x_{\tau\setminus 2}$	28	-7.92E+6	1.5E+2

In order to illustrate the tendency of the proposed algorithm to avoid local-nonglobal minimizers, we compare Algorithm 3.1 with the well-known limited memory algorithm for solving large nonlinear unconstrained (or box constrained) optimization problems[1] (L-BFGS-B); see [5]. Focusing on the amount of storage required by L-BFGS-B, the user should set a parameter m that determines the number of BFGS corrections to be saved. For our experiments, we set $m = p$ in order to use a similar amount of storage as in Algorithm 3.1, and we use all the default parameters for L-BFGS-B. On Table 2 we present the obtained results for both algorithms applied to the multidimensional separable function (85), setting the initial point much closer to x_l than to x_τ such that each component is given by a random number with uniform distribution in $[-1.5, -0.5]$ for the odd entries, and equal to -2.0 for the even entries. We can observe that Algorithm 3.1 has a clear tendency to escape from the neighborhood of x_l, and to converge to a local minimizer for which the objective value is much lower than the one obtained by the limit point of the sequence generated by the L-BFGS-B strategy.

For our second example, we consider the multi-dimensional nonseparable quartic function:

$$\left((x_1 - 2)^2 + 10 \sum_{i=2}^{n} {x_i}^2 + 10 \left(x^T x - 1 \right)^2 \right). \tag{86}$$

[1] available at `http://users.iems.northwestern.edu/ nocedal/lbfgsb.html`

Table 2: Performance of LBFGS-B and Algorithm 3.1 when applied to (85).

		L-BFGS-B		Algorithm 3.1	
n	p	x^*	f^*	x^*	f^*
400	7	$x_{\tau\backslash 200}$	8.2E+3	$x_{\tau\backslash 160}$	-1.0E+5
	10	$x_{\tau\backslash 200}$	8.2E+3	$x_{\tau\backslash 180}$	-5.3E+4
	15	$x_{\tau\backslash 200}$	8.2E+3	$x_{\tau\backslash 154}$	-7.6E+4
1000	10	$x_{\tau\backslash 500}$	4.8E+4	$x_{\tau\backslash 459}$	-2.7E+5
	15	$x_{\tau\backslash 500}$	4.8E+4	$x_{\tau\backslash 383}$	-5.5E+5
	50	$x_{\tau\backslash 500}$	4.8E+4	$x_{\tau\backslash 298}$	-7.3E+5

For any n, this function has a local minimum at $x_l \simeq (-0.917, 0, \ldots, 0)^T$, a unique global minimum at $x_\tau \simeq (1.023, 0, \ldots, 0)^T$ and a local maximum near the origin [24]. Although in [24] the dimension of the problems were smaller we used some of the initial guesses suggested therein for our experiments, as well as some other initial choices. For that, in Tables 3 and 4, x_0 denotes the initial point and $x_i \sim \mathrm{U}[a, b]$ stands for a random entry of x_0 with uniform distribution in $[a, b]$. Algorithm 3.1 has shown a good performance with this function, for different values of n, since the use of a very small subspace was enough to illustrate its convergence behavior. We present in Table 3, for a medium size case of function (86), the comparison between Algorithm 3.1 and a hard-case-free separable trust-region Newton-Lanczos strategy, which is based on a quadratic model that is also embedded in our proposal by considering $\widehat{\rho}_k^i = 0$ in (80), for all k and all i. We can observe that, regardless of the initial point, the sequence generated by the cubic model converges quite frequently to the global minimizer, at the cost of some possible additional iterations, as compared with the sequence generated by the quadratic model that converges in most cases to the local minimizer.

Table 3: Cubic versus quadratic model, from Algorithm 3.1, when applied to (86) for $n = 500$ and $p = 3$.

	Cubic model		Quadratic model	
x_0^T	x^*	it	x^*	it
$(0, 0, \ldots, 0)$	x_l	6	x_l	6
$(-0.75, 0.01, 0, \ldots, 0)$	x_l	17	x_l	9
$(-0.2, 0.01, 0, \ldots, 0)$	x_τ	11	x_l	15
$(-0.01, 1.1, 0, \ldots, 0)$	x_τ	14	x_l	11
$(1.0, 0, \ldots, 0)$	x_τ	4	x_τ	4
$x_i \sim \mathrm{U}[-1, 1]$, $\forall i$	x_τ	30	x_l	14
$x_i \sim \mathrm{U}[-2, 2]$, $\forall i$	x_τ	27	x_l	21
$x_i \sim \mathrm{U}[-0.05, 0.05]$, $\forall i$	x_τ	17	x_l	16
$x_i \sim \mathrm{U}[0, 1], \forall i \neq 1,\ x_1 = -0.917$	x_τ	22	x_l	17

To report the behavior of our proposal for larger values of n, we also report on Table 4 the results obtained for $n = 5000$ and $p = 2$ when applied to (86). Once again, we can observe that, regardless of the initial point, the sequence generated by the cubic model converges quite frequently to the global minimizer.

The remainder sets of test instances are related to the disk packing problem. This problem, which is simple to describe but very hard to solve, consists in placing q circles of radius r into a rectangular box $[0, d_1] \times [0, d_2]$ in such a way that the intersection between any pair of circles i and j, $i \neq j$, is at most one point, i.e., the circles are not overlapped. Consider $p^i = (p_1^i, p_2^i)$, $i = 1, \ldots, q$, the centers of the desired circles. Therefore, the packing problem can

Table 4: Performance of Algorithm 3.1 when applied to (86) for $n = 5000$ and $p = 2$.

x_0^T	x^*	IT	x_0^T	x^*	IT	x_0^T	x^*	IT
$(0, 0, \ldots, 0)$	x_l	6	$(0.01, -2.0, 0, \ldots, 0)$	x_τ	10	$(-0.1, 3.5, 0, \ldots, 0)$	x_τ	19
$(-0.75, 0.01, 0, \ldots, 0)$	x_l	6	$(-0.05, 0, -0.05 \ldots, -0.05)$	x_τ	19	$(-0.01, -2.0, 0, \ldots, 0)$	x_τ	10
$(0.5, 0.2, 0, \ldots, 0)$	x_τ	7	$x_i \sim$ U$[-1, 1]$, $\forall i$	x_τ	41	$(-0.2, 0.01, 0, \ldots, 0)$	x_l	7
$(-0.01, 1.1, 0, \ldots, 0)$	x_τ	11	$x_i \sim$ U$[-2, 2]$, $\forall i$	x_τ	22	$(-0.917, 0.06, \ldots, 0.06)$	x_τ	22
$(1.0, 0, \ldots, 0)$	x_τ	4	$x_i \sim$ U$[-0.05, 0.05]$, $\forall i$	x_τ	10	$(0.01, -1.1, 0, \ldots, 0)$	x_τ	11
$(-0.2, 1.1, 0, \ldots, 0)$	x_τ	11	$x_i \sim$ U$[0, 1], \forall i \neq 1,\ x_1 = -0.917$	x_τ	41	$(0, -0.01, 0 \ldots, 0)$	x_τ	8

be formulated as follows [3]

$$\begin{aligned} \min \quad & \sum_{i \neq j} (\min(0, ||p^i - p^j||_2^2 - (2r)^2))^2 \\ \text{subject to} \quad & r \leq p_1^i \leq d_1 - r, \quad i = 1, \ldots, q, \\ & r \leq p_2^i \leq d_2 - r, \quad i = 1, \ldots, q. \end{aligned} \tag{87}$$

As usual in optimization, this problem can be rewritten as an unconstrained problem by incorporating the constraints into the objective function as follows:

$$\min \ f(x) = \sum_{i \neq j} (\min(0, ||p^i - p^j||_2^2 - (2r)^2))^2 + \sum_{i=1}^{q} \sum_{l=1}^{2} (\min(0, p_l^i - r))^2 + (\min(0, d_l - r - p_l^i))^2. \tag{88}$$

Despite its simplicity, the function $f(x)$ has a large number of local-nonglobal minimizers. Nevertheless, if $f(\bar{x}) = 0$ then $\bar{x}$ is a global minimizer, and hence it is a solution of the disk packing problem.

In our experiments, we solved problem (88) where q, r, d_1 and d_2 are given positive numbers. For that, we ran Algorithm 3.1 for different initial guesses $p^i \in \mathbb{R}^2, i = 1, \ldots, q$ randomly generated from a uniform distribution inside the feasible region of problem (87). We tested various dimensions of the subspace $2 \leq p \leq q$ in our experiments and, for each value of p, a different initial guess was generated. If the algorithm did not find a solution of the packing problem with the combination of the initial guess generated by the random generator and the dimension of the subspace, considering a provided seed, we modified the seed and the process was repeated until a maximum of thirty seeds had been tested or a solution was computed. Next we describe the steps performed for our experiments.

Procedure of experiments

set $j = 1$, $fbest = 10^8$ and $tol = 10^{-6}$.
while ($fbest > tol$ and $j \leq 30$) **do**
  set seed = seed$_j$ and $p = 2$.
  while ($fbest > tol$ and $p \leq q$) **do**
    Generate a random initial guess $p^i \in \mathbb{R}^2, i = 1, \ldots, q$.
    Run **Algorithm 3.1** and compute f.
    if ($f < fbest$) **then**
      update $fbest \leftarrow f$.
    end if
    $p \leftarrow p + 1$

end while

$j \leftarrow j + 1$

end while

We solved the packing problems described in [10, 11] that were also tested in [3] with improvements for some instances. Table 5 presents the performance of the algorithm to solve these instances, where the notation SEED stands for the total number of seeds used to obtain a solution.

Table 5: Performance of algorithm to solve problems in [3].

PROBLEM	$d_1 \times d_2$	q	r	SEED	p	IT	TIME
1.1	160 × 80	91	6	1	29	76966	1.19E+2
1.2	100 × 200	84	8	1	27	73766	9.43E+1
1.3	120 × 240	74	10	1	51	133639	2.01E+2
1.4	100 × 80	86	5	1	25	53711	6.72E+1
1.5	120 × 80	68	6	1	19	51859	4.31E+1
1.6	120 × 100	87	6	1	39	86467	1.28E+2
1.7	80 × 80	68	5	1	20	46376	3.63E+1
1.8	100 × 100	71	6	1	42	100491	1.17E+2
1.9	120 × 120	74	7	1	20	55813	5.38E+1
2.1	160 × 80	32	10	1	24	60115	1.74E+1
2.2	100 × 200	29	13	1	25	66324	1.72E+1
2.3	120 × 240	32	15	1	14	38593	8.34E+0
2.4	100 × 80	32	8	1	15	39566	9.11E+0
2.5	120 × 80	30	9	1	16	43030	9.07E+0
2.6	120 × 100	30	10	2	11	93947	2.31E+1
2.7	80 × 80	32	7	1	19	52818	1.36E+1
2.8	100 × 100	30	9	1	14	37221	7.28E+0
2.9	120 × 120	30	11	1	21	59577	1.43E+1
3.1	160 × 80	15	14	1	10	26440	1.62E+0
3.2	100 × 200	15	18	1	13	34352	2.43E+0
3.3	120 × 240	15	21	1	15	41446	3.21E+0
3.4	100 × 80	16	11	1	12	31219	2.37E+0
3.5	120 × 80	15	12	1	9	22951	1.36E+0
3.6	120 × 100	14	14	1	11	28476	1.69E+0
3.7	80 × 80	16	10	1	6	14137	8.27E-1
3.8	100 × 100	13	13	20	13	675676	4.10E+1
3.9	120 × 120	16	15	1	7	17692	1.08E+0
4.1	160 × 80	8	20	1	6	12791	2.61E-1
4.2	100 × 200	8	25	1	5	11956	2.38E-1
4.3	120 × 240	8	30	1	6	13258	2.70E-1
4.4	100 × 80	6	16	1	3	4581	4.80E-2
4.5	120 × 80	7	17	1	5	10882	1.79E-1
4.6	120 × 100	6	20	1	4	8795	1.10E-1
4.7	80 × 80	6	14	1	4	7118	8.60E-2
4.8	100 × 100	6	18	1	6	11455	1.52E-1
4.9	120 × 120	6	21	1	6	13100	1.89E-1
5.1	160 × 80	3	25	1	3	3405	1.50E-2
5.2	100 × 200	3	31	1	2	11	0.00E+0
5.3	120 × 240	3	37	1	2	325	2.00E-3
5.4	100 × 80	4	19	1	3	4705	3.00E-2
5.5	120 × 80	4	21	1	4	3560	2.90E-2
5.6	120 × 100	4	24	1	2	2885	1.60E-2
5.7	80 × 80	4	17	1	2	3000	1.60E-2
5.8	100 × 100	4	22	1	3	3248	1.90E-2
5.9	120 × 120	4	26	1	3	3913	2.40E-2

The numerical results indicate that Algorithm 3.1 was efficient to find a global solution of

the packing problem (88), obtaining in all cases at convergence an objective function value of the order of 10^{-6}. Furthermore, the algorithm required a small amount of effort for the smallest instances (problems from subsets 3.x, 4.x and 5.x) but this effort increases when the dimension of the problems increases. However, it is important to notice that the dimension of the subspace in each instance is quite small when compared to the dimension of the problem, specially for the biggest dimensions. Moreover, the smaller problems needed dimensions of the subspace closer to the number of points and for the biggest dimensions (problems 1.x) the dimension of the subspace did not increase meaningfully. For this reason, besides the higher number of iterations, the time spent to solve the problem is relatively small.

Our next set of test problems derived from the Packomania collection [29], where the maximum radius (best known) for packing equal circles in a square of side one is reported. In particular, for $q = m^2$ and $m \in \{2, \ldots, 6\}$, the regular grid $m \times m$ has been geometrically proved to be the global minimum of the problem. For $m = 7$ it was computationally shown that the maximum radius is strictly larger than the radius $1/14$ associated to the regular grid. We ran several experiments increasing the number of disks, starting with $q = 49$. Our experiments for these instances clearly showed a better performance of the algorithm, concerning number of iterations and CPU time, when low dimensional subspaces were used. Therefore, we report in Table 6 the performance results when the maximum dimension p of the subspace was set to 15. In particular, in Figure 1 we show the obtained solution for the instance with 49 circles.

Table 6: Instances from Packomania collection [29].

q	r	SEED	p	IT	TIME
49	7.169268E-2	2	6	4097	1.98E+0
64	6.345899E-2	2	15	10683	7.31E+0
81	5.686992E-2	2	11	10984	1.07E+1
100	5.140107E-2	2	13	12630	1.88E+1
400	2.620235E-2	1	15	9604	2.45E+2
1020	1.653411E-2	9	15	114179	1.82E+4
2015	1.160255E-2	1	14	7616	4.69E+3

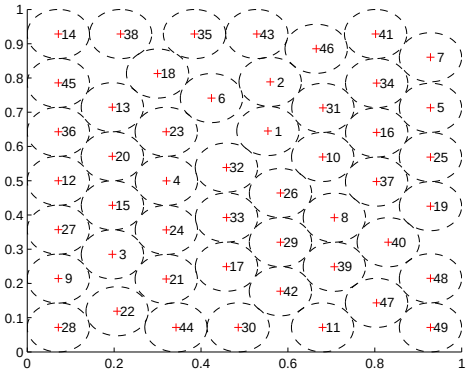

Figure 1: 49 circles of radius 7.169268E-02 packed in a square of side one.

Despite the high number of iterations, the algorithm was able to find a global solution in a reasonable amount of time taking into account the dimension and difficulty of these disk packing

problems. As stated before, the computational effort is small for the smallest problems but tends to increase with the dimension of the problems. However, the algorithm found a solution for all the tested problems and the dimension of the subspace was quite small.

In order to have a better idea of the efficiency of the algorithm, we compared its performance on the packing instances with the global optimization solver GAMS-BARON [33]. Since the objective function of problem (87) has discontinuous derivatives, it is required to use the DNLP (Discontinuous Nonlinear Program) model option when applied to (87), which is strongly recommendable to avoid[2]. Therefore, we considered a global optimization formulation of the packing problem that incorporates the non-overlapping condition of the circles into the constraints and leads to a problem with a smooth function and nonconvex constraints. The formulation is the following:

$$\begin{array}{rll} \min & \displaystyle\sum_{i=1}^{q} p_1^i + p_2^i & \\ \text{subject to} & r \leq p_1^i \leq d_1 - r, & i = 1, \ldots, q, \\ & r \leq p_2^i \leq d_2 - r, & i = 1, \ldots, q, \\ & ||p^i - p^j||_2^2 \geq (2r)^2, & i \neq j. \end{array} \tag{89}$$

This new formulation of the packing problem intends to favor the performance of any solver suitable to deal with smooth nonlinear programs, like BARON. In fact, we can observe that a feasible solution for this problem is a feasible solution of problem (87). Moreover, the third set of constraints in (89) guarantees that any feasible solution satisfies $\min(0, ||p^i - p^j||_2^2 - (2r)^2) = 0, \forall i \neq j$, which is an interesting feature that implies a zero value for the objective function in (87). On the other hand, since the objective function of problem (87) is bounded below by zero, any feasible solution with null objective value is a global solution. The aforementioned characteristics allow us to conclude that a feasible solution of (89) is a global solution of the disk packing problem. Therefore, since the objective function of (89) is irrelevant to the achievement of the global solution, we have chosen a trivial linear function to favor the performance of the global solver BARON in finding a solution of the packing problem. For this reason, in our experiments with BARON we included the solver options to force the termination search once a feasible solution is found or a maximum of 2.0E+04 seconds of execution is attained. Despite all this favorable features of (89) for the performance of BARON, the solver was only able to find a feasible solution for one of the problems from Table 6, within the allowed CPU time (the instance with 81 circles was solved in 2.14E+02 seconds).

It is important to mention that BARON solved effectively all the instances of smaller dimensions ($q = m^2$ and $m \in \{2, \ldots, 7\}$) in the preprocessing phase when the radius corresponds to the regular grid but, for example, in the instance with 49 circles ($m = 7$), a slight increase in the radius ($r = 7.169268\text{E-}02$) induces difficulties to find a feasible solution. These results illustrate the advantages of employing Algorithm 3.1 to solve the disk packing problem instead of a well-known and established global optimizer.

5 Final remarks

We introduced a trust-region method with subspace minimization for solving large-scale unconstrained minimization problems. At each iteration we minimize a model that may coincide with

[2] see, e.g., `https://www.gams.com/latest/docs/UG_ModelSolve.html`

the objective function up to first or second order, giving different convergence and complexity properties. The model used in practice was a second-order approximation of the objective function plus an empirical approximation of third-order Taylor terms. The specific choice of this model, its minimization onto small-dimensional subspaces, and the use of a variable trust-region norm make it possible the model minimization in linear time. For choosing the small-dimensional subspaces we use Lanczos method, which, in general, makes it possible the fast detection of negative curvature directions, improving the approximation to second-order stationary points. Some numerical experiments show that the new method is able to handle the minimization of nonconvex functions with many variables.

Future research involves: applications to machine learning and statistical learning; a comprehensive comparison with well-established large-scale minimization algorithms; and improving the secant-like strategies employed to approximate the third-order tensor.

Acknowledgements. We would like to thank two anonymous referees for their constructive comments and suggestions that helped us to improve the final version of this paper.

References

[1] E. Bergou, Y. Diouane, and S. Gratton [2017], *On the use of the energy norm in trust-region and adaptive cubic regularization subproblems*, Computational Optimization and Applications, 68, pp. 533–554.

[2] E. Bergou, Y. Diouane, and S. Gratton [2018], *A line-search algorithm inspired by the adaptive cubic regularization framework and Complexity Analysis*, Journal of Optimization Theory and Applications, 178, pp. 885–913.

[3] E. G. Birgin, J. M. Martínez, and D. P. Ronconi [2005], *Optimizing the packing of cylinders into a rectangular container: a nonlinear approach*, European Journal of Operational Research, 160, pp. 19–33.

[4] P.N. Brown, H. F. Walker, R. Wasyk, and C. S. Woodward [2008], *On using approximate finite-differences in matrix-free Newton-Krylov methods*, SIAM Journal on Numerical Analysis, 46, pp. 1892–1911.

[5] R. H. Byrd, P. Lu, and J. Nocedal [1995], *A Limited Memory Algorithm for Bound Constrained Optimization*, SIAM Journal on Scientific and Statistical Computing, 16, pp. 1190–1208.

[6] C. Cartis, N. I. M. Gould, and Ph. L. Toint [2010], *On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization*, SIAM Journal on Optimization, 20, pp. 2833–2852.

[7] C. Cartis, N. I. M. Gould, and Ph. L. Toint [2011], *Adaptive cubic regularization methods for unconstrained optimization. Part I: motivation motivation, convergence and numerical results*, Mathematical Programming, 127, pp. 245–295.

[8] C. Cartis, N. I. M. Gould, and Ph. L. Toint [2011], *Adaptive cubic regularization methods for unconstrained optimization. Part II: worst-case function and derivative complexity*, Mathematical Programming, 130, pp. 295–319.

[9] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Trust Region Methods* [2000], Society for Industrial and Applied Mathematics, Philadelphia.

[10] M.H. Correia, J.F. Oliveira, and J.S. Ferreira [2000], *Cylinder packing by simulated annealing*, Pesquisa Operacional, 20, pp. 269–284.

[11] M.H. Correia, J.F. Oliveira, J.S. Ferreira [2001], *A new upper bound for the cylinder packing problem*, International Transactions in Operational Research, 8, pp. 571–583.

[12] F. E. Curtis, D. P. Robinson, and M. Samadi [2017], *A trust-region algorithm with a worst-case iteration complexity of $O(\varepsilon^{-3/2})$*, Mathematical Programming, 162, pp. 1–32.

[13] J. W. Demmel [1997], *Applied Numerical Linear Algebra*, SIAM, Philadelphia.

[14] G. Fasano and S. Lucidi [2009], *A nonmonotone truncated Newton-Krylov method exploiting negative curvature directions, for large scale unconstrained optimization*, Optimization Letters, 3, pp. 521–535.

[15] G. Fasano and M. Roma [2013], *Preconditioning Newton-Krylov methods in nonconvex large scale optimization*, Computational Optimization and Applications, 56, pp. 253–290.

[16] N.I.M. Gould, S. Lucidi, M. Roma, and Ph.L. Toint [1999], *Solving the trust-region subproblem using the Lanczos method*, SIAM Journal on Optimization, 9, pp. 504–525.

[17] G. N. Grapiglia and Y. Nesterov [2017], *Globally convergent second-order schemes for minimizing twice differentiable functions*, CORE Discussion Paper 2016/28, Université Catholique de Louvain, Louvain, Belgium.

[18] G. N. Grapiglia, J.-Y. Yuan, and Y.-X. Yuan [2015], *On the convergence and worst-case complexity of trust-region and regularization methods for unconstrained optimization*, Mathematical Programming, 152, pp. 491–520.

[19] A. Griewank [1981], *The modification of Newton's method for unconstrained optimization by bounding cubic terms*, Technical Report NA/12, Department of Applied Mathematics and Theoretical Physics, University of Cambridge.

[20] L. Komzsik [2003], *The Lanczos Method: Evolution and Application*, SIAM, Philadelphia.

[21] C. Lanczos [1950], *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Standards, 45, pp. 255–282.

[22] LAPACK–Linear Algebra PACKage [2013], `http://www.netlib.org/lapack`.

[23] S. Lu, Z. Wei, and L. Li [2012], *A trust region algorithm with adaptive cubic regularization methods for nonsmooth convex minimization*, Computational Optimization and Applications, 51, pp. 551–573.

[24] J.M. Martínez and M. Raydan [2015], *Separable cubic modeling and a trust-region strategy for unconstrained minimization with impact in global optimization*, Journal of Global Optimization, 63(2), pp. 319–342.

[25] J.M. Martínez and M. Raydan [2017], *Cubic-regularization counterpart of a variable-norm trust-region method for unconstrained minimization*, Journal of Global Optimization, 68, pp. 367–385.

[26] J. J. Moré and D. C. Sorensen [1983], *Computing a trust region step*, SIAM Journal on Scientific and Statistical Computing, 4, pp. 553–572.

[27] S.G. Nash [1984], *Newton-type minimization via the Lanczos method*, SIAM Journal on Numerical Analysis, 21, pp. 770–788.

[28] Y. Nesterov and B. T. Polyak [2006], *Cubic reglarization of Newton's method and its global performance*, Mathematical Programming, 108, pp. 177–205.

[29] Packomania. Test instances available at `http://www.packomania.com/`, last accessed 3 March 2018.

[30] M. Rojas, S.A. Santos, and D.C. Sorensen [2000], *A New Matrix-Free Algorithm for the Large-Scale Trust-Region Subproblem*, SIAM Journal on Optimization, 11, pp. 611–646.

[31] Y. Saad [2011], *Numerical Methods for Large Eigenvalue Problems*, 2nd Edition, SIAM, Philadelphia.

[32] G. W. Stewart [2001], *Matrix Algorithms: Volume II. Eigensystems*, SIAM, Philadelphia.

[33] M. Tawarmalani and N. V. Sahinidis [2005], *A polyhedral branch-and-cut approach to global optimization*, Mathematical Programming, 103(2), pp 225–249.

[34] Z.-H. Wang and Y.-X. Yuan [2006], *A subspace implementation of quasi-Newton trust region methods for unconstrained optimization*, Numerische Mathematik, 104, pp. 241–269.

[35] Y. Yuan [2000], *On the truncated conjugate gradient method*, Mathematical Programming, 87, pp. 561–571.