# Composite Proximal Bundle Method

Claudia Sagastizábal*

**Abstract**

We consider minimization of nonsmooth functions which can be represented as the composition of a positively homogeneous convex function and a smooth mapping. This is a sufficiently rich class that includes max-functions, largest eigenvalue functions, and norm-1 regularized functions. The bundle method uses an oracle that is able to compute separately the function and subgradient information for the convex function, and the function and derivatives for the smooth mapping. With this information, it is possible to solve approximately certain proximal linearized subproblems in which the smooth mapping is replaced by its Taylor-series linearization around the current serious step. Our numerical results show the good performance of the Composite Bundle method for a large class of problems.

## 1 Introduction and motivation

For some years already, nonsmooth optimization research has focused on exploiting structure in the objective function as a way to speed up numerical methods. Indeed, for convex optimization, complexity results establish that oracle based methods have a linear rate of convergence; [NY83]. The $\mathcal{U}$-Lagrangian theory [LS97], [LOS00], and the $\mathcal{VU}$-space decomposition [MS00], can be seen as tools to "extract" smooth structure from general convex functions. The approach was extended to a class of nonconvex functions in [MS04]. Similar ideas for more general nonconvex functions, having a smooth representative on a certain manifold corresponding to the $\mathcal{U}$-subspace, are explored in [Lew02], [Har03]. Another line of work concentrates efforts on identifying classes of functions structured enough to have some type of second-order developments, such as the primal-dual gradient structured functions from [MS00], [MS03], or the composite functions in [Sha03]. Composite functions, that were first considered in [Fle87, Ch. 14], and studied in [BF95], [LW02], have been more recently revisited in [Nes07] and [LW08], in the convex and nonconvex settings, respectively.

Most of the work above is conceptual, in the sense that if some algorithmic framework is considered, often it is not implementable. Among the exceptions, we find the convex optimization algorithms in [Ous00], [MS05], [Nes07], and the $\mathcal{V}$-space identification method in [DSS09]. In this paper, we develop an implementable bundle method that exploits structure for certain nonconvex composite optimization problems. More precisely, given a smooth mapping $c : \Re^n \to \Re^m$ and a positively homogeneous (of degree 1) convex function $h : \Re^m \to \Re$, we consider how to solve the unconstrained problem

$$\min_{x\in\Re^n} (h \circ c)(x)\,. \tag{1}$$

In this composite objective, we refer to $c$ as the *inner* mapping and to $h$ as the *outer* function. Note in particular that the outer function is real-valued on $\Re^m$, an assumption that excludes indicator functions, but still covers a rich enough family of interesting problems; see § 2.

For solving the possibly nonconvex problem (1) we develop a specialized bundle method that takes full advantage of the composite structure of the objective function. Specifically, rather than using composite data:

$$(\mathtt{bb}_{\mathtt{hoc}}):\quad \text{Given } x\in\Re^n,\quad \text{a composite black-box computes}\quad (h\circ c)(x) \text{ and } \gamma\in\partial(h\circ c)(x)\,,$$

*CEPEL, Electric Energy Research Center, Eletrobrás Group. On leave from INRIA Rocquencourt, France. sagastiz@impa.br

we suppose the oracle has the ability to make separate computations. More precisely,

$$\begin{cases} (\mathtt{bb_c}): & \text{Given } x \in \Re^n \quad \text{an inner black-box computes} \quad c(x) \text{ and its Jacobian } Dc(x) \\ (\mathtt{bb_h}): & \text{Given } C \in \Re^m \quad \text{an outer black-box computes} \quad h(C) \text{ and a subgradient } G \in \partial h(C). \end{cases}$$

While $(\mathtt{bb}_{h \circ c})$ would build a classical cutting-planes model for the composite objective, having the richer oracle $(\mathtt{bb_c})-(\mathtt{bb_h})$ at hand, we can make a better approximation and build a cutting-planes model for the so-called *conceptual model*. This model, introduced in [LW08] for general composite functions, and given by (4) below, defines certain proximal linearized subproblems in which the smooth mapping is replaced by its Taylor-series linearization. The corresponding algorithmic framework, named ProxDescent in [LW08], is rather broad (outer functions can be extended-valued and prox-regular), but relies heavily on the exact computation of proximal points at each iteration. In this work, we provide a fully implementable variant of such algorithmic framework for a less general, but still large enough, class of outer functions (real-valued, positively homogeneous, and convex).

In order to deal with approximate proximal computations, our fully implementable variant, given by Algorithm 1 below, incorporates typical features from bundle methods, such as *serious* and *null* steps iterations. But this is not the only modification of ProxDescent: there is an additional level of approximation, introduced by the need to estimate the conceptual model, which is handled in the implementable algorithm by backtracking. For these reasons, our convergence analysis deviates significantly from that of [LW08]. In addition, instead of a standard proximal term as in ProxDescent, in our development we consider a variable prox-metric that opens the way to exploiting second-order information of the smooth mapping, when available.

An alternative line of work, based on somewhat different richer oracles (that can compute more than one subgradient at any given point), refers to the proximity control bundle method [NPR08]. These methods, devised for nonconvex optimization problems appearing in eigenvalue optimization and automatic control, use certain "first-order local models" that contain the conceptual model from [LW08] as a special case.

Our paper is organized as follows. In Sec. 2 we give several classes of functions with composite structure. The conceptual model and the proposed cutting-planes function, that makes the conceptual model computationally tractable, are described in Sec. 3. Sections 4 and 5 contain, respectively, the Composite Bundle method and its convergence results. In Sec. 6 we report our numerical experience, on many of the functions described in Sec. 2. Our results show the good performance of the composite algorithm, when compared to several solvers, on a large number of problems of various dimensions. In the final section with concluding remarks, we compare the main features of our composite method with the proximity control bundle method from [NPR08].

# 2 Composite functions in the family

Our class of composite functions includes many examples from [LW08], [Nes07], [Sha03]. We mention in passing that the wording "composite" means slightly different things for different authors. Specifically, while we refer to a composition that can be nonconvex (as in [LW08], [Sha03]), composite functions in [Nes07] are the convex sum of two terms, one convex and another one smooth.

We suppose the outer function is positively homogeneous and convex, so it is also sublinear. In [HUL01, Lesson C] there are many examples of real-valued sublinear functions, such as norms, quadratic seminorms, gauges of closed convex sets containing 0 in their interior, infimal convolutions of sublinear functions, and support functions of bounded sets. The composition of any of such (outer) functions with any smooth (inner) mapping defines a function in our family. We review below some interesting special cases.

**Example 1 (max-functions)** The function given by the pointwise maximum of a finite collection of $C^2$-functions in $\Re^n$, $\{c_1(\cdot), \dots, c_m(\cdot)\}$, can be defined by composing the outer function $h(C) = \max(C_1, \dots, C_m)$ with the inner mapping with components $c_j(\cdot)$, for $j = 1, \dots, m$.

For any $x \in \Re^n$, the Jacobian mapping $Dc(x)$ is the $m \times n$ matrix with rows given by the transposed gradients $\nabla c_j(x)^\top$. As for the second-order derivative, $D^2c(x)$, for any $d, \tilde{d} \in \Re^n$, $[D^2c(x)\tilde{d}]d$ is a vector in $\Re^m$ with $j^{th}$ component given by $\tilde{d}^\top \nabla^2 c_j(x) d$.

We now give a more involved function, appearing in semidefinite programming.

**Example 2 (eigenvalue optimization)** Let $\mathcal{S}^p$ denote the linear space of symmetric matrices of order $p$, and consider the function $\lambda_{\max}(X)$, given by the maximum eigenvalue of a symmetric matrix $X \in \mathcal{S}^p$, whose elements are $C^2$-functions of a vector $x \in \Re^n$. Suppose that for a fixed $X$, $r$ denotes the multiplicity of the maximum eigenvalue. Then from the analysis in [BS00, Ex. 3.98], the orthogonal projection onto the eigenspace corresponding to the maximum eigenvalue is an analytic function near $X$. By applying Gram-Schmidt orthonormalization to the columns of such projection, it is possible to define a "diagonalizing" analytic mapping $\mathcal{C} : \mathcal{S}^p \to \mathcal{S}^r$, such that: $\mathcal{C}(X) = \lambda_{\max}(X)\mathrm{I}_r$ with $\mathrm{I}_r$ the identity matrix of order $r$; the Jacobian $D\mathcal{C}(X)$ is surjective; and the eigenvalues $\lambda(\mathcal{C}(X))$ coincide with $\lambda_{\max}(X)$. The inner smooth mapping corresponds to $\mathcal{C}$, while the outer function is the eigenvalue function $h(C) = \lambda(C)$, a positively homogeneous convex function.

The next function is typically used in applications such as wavelet-based image restoration, sparse representations, sparse regression, and compressive sensing problems.

**Example 3 (Regularized minimization maps)** Many signal or image reconstruction problems, as well as approximation problems, minimize a function of the form

$$\frac{1}{2}|Ax - y|^2 + \tau|x|\,,$$

for $x \in \Re^n$, $y \in \Re^k$, $A$ a matrix $k \times n$, $\tau$ a positive parameter, and for given appropriate norms. In the expression above, the first *data fidelity* term is smooth, and the second term corresponds to some regularization or penalty, for example ensuring sparsity of the solution. To fit our composite framework, it suffices to take $m = n + 1$,

$$c_j(x) = x_j \ \text{ for } j = 1, \ldots, n\,, \qquad c_{n+1}(x) = \frac{1}{2}|Ax - y|^2\,;$$

and define the outer function $h(C_1, \ldots, C_{n+1}) = C_{n+1} + \tau|(C_1, \ldots, C_n)|$.

For any $x \in \Re^n$, the Jacobian mapping $Dc(x)$ is the $(n+1) \times n$ matrix made by appending to the identity matrix of order $n$ a row with the transpose of $A^\top(Ax - y)$. As for the second-order derivative, $D^2c(x)$, for any $d, \tilde{d} \in \Re^n$, $[D^2c(x)\tilde{d}]d$ is a vector with its first $n$ components equal to 0, its last component equal to $\tilde{d}^\top A^\top Ad$, a constant with respect to $x$.

We now give a possibly nonconvex function, appearing in approximation problems.

**Example 4 (sum of Euclidean norms)** Given a collection of smooth vector functions, $\{\phi_1, \ldots, \phi_J\}$ with $\phi_j : \Re^n \to \Re^{m_j}$ for $j = 1, \ldots, J$, the function $\sum_{j=1}^J |\phi_j(x)|$ is the composition of the following smooth mapping with $m = \sum_{j=1}^J m_j$ components, and outer function:

$$c(x) = \Big(\phi_1(x), \ldots, \phi_J(x)\Big) \quad \text{and} \quad h(C_1, \ldots, C_{m_1}, C_{m_1+1}, \ldots, C_m) = \sum_{j=1}^J \Big|(C_{\sum_{k=1}^{j-1} m_k + 1}, \ldots, C_{\sum_{k=1}^{j} m_k})\Big|\,.$$

When $n = J = m_1 = 1$, letting $\phi_1(x) = 0.5a_2x^2 + a_1x + a_0$ for $a_{2,1,0}$ given scalars, it is easy to see that the function $f$ is convex if $a_2 > 0$ with $a_1^2 < 2a_0a_2$, and concave otherwise.

The next function shows how to cast $\ell_1$-penalizations of nonlinear programming problems into the composite framework (provided no indicator function is used for polyhedral sets, to preserve finite-valuedness of the outer function).

**Example 5 (Uryasev's exact penalty function)** This is an optimal control problem over a discrete horizon with $n$ time steps. The control variable $x \in \Re^n$ is constrained to the box $[-0.2, 0.2]^n$. Given initial values $(\xi_0, \psi_0)$, the state variables $(\xi, \psi) \in \Re^2$ follow a trajectory given by a recursive state equation of the form

$$\begin{array}{ll} \xi_1(x_1) = \xi_0 + 0.2\psi_0\,, & \psi_1(x_1) = -0.004 + 0.2x_1, \text{ and} \\ \xi_{i+1}(x_{1:i+1}) = \xi_i(x_{1:i}) + 0.2\psi_i(x_{1:i})\,, & \psi_{i+1}(x_{1:i+1}) = \psi_i(x_{1:i}) - \texttt{cubic}\,\psi_i(x_{1:i})^2 - 0.004\xi_i(x_{1:i}) + 0.2x_{i+1}\,, \end{array}$$

for $i = 1, \ldots, n-1$ and where `cubic` $\geq 0$ is a given parameter. In the expressions above, we use the notation $x_{1:i}$ to refer to the dependence of the $i^{th}$-state variables on the first $i^{th}$ components of the control variable $x$. The control problem to be solved is:

$$\begin{cases} \min_x & \frac{1}{2}\sum_{i=1}^n \xi_i(x_{1:i})^2 \\ & x_i \in [-0.2, 0.2] \quad \text{for } i = 1, \ldots, n \\ & \psi_i(x_{1:i}) \geq -1 \quad \text{for } i = 1, \ldots, n-1 \\ & \psi_n(x_{1:n}) = 0\,. \end{cases}$$

The corresponding $\ell_1^+$-penalty function uses parameters $\mathtt{c}_1$, $\mathtt{c}_2$, $\mathtt{c}_3 > 0$, and can be written in composite form by letting $m = 3n+1$,

$$c_j(x) = \begin{cases} -1 - \psi_j(x_{1:j}) & j = 1, \ldots, n-1 \\ \psi_n(x_{1:n}) & j = n \\ \frac{1}{2}\sum_{i=1}^n \xi_i(x_{1:i})^2 & j = n+1 \\ x_{j-n-1} - 0.2 & j = n+2, \ldots, 2n+1 \\ -x_{j-2n-1} - 0.2 & j = 2n+2, \ldots, 3n+1\,, \end{cases}$$

and taking the outer function

$$h(C) = C_{n+1} + \mathtt{c}_1\Big(|C_{n+2:2n+1}|^+ + |C_{2n+2:3n+1}|^+\Big) + \mathtt{c}_2|C_n| + \mathtt{c}_3|C_{1:n-1}|^+\,.$$

The Jacobian mapping $Dc(x)$ can be computed using the adjoint state equations, to obtain the trajectory derivatives $\nabla\xi_i$ and $\nabla\psi_i$. When the parameter `cubic` is null, the control problem is linear-quadratic and the composite function $h \circ c$ is convex. In this case, the second-order derivative, $[D^2c(x)\tilde{d}]d$ is a vector with all of its components equal to 0, except for the $(n+1)^{th}$ component, equal to $\tilde{d}^\top \sum_{i=1}^n \nabla\xi_i \nabla\xi_i^\top d$ for any $d, \tilde{d} \in \Re^n$. When `cubic > 0`, the function is nonconvex, and second-order derivatives are difficult to compute for higher dimensions. For $n = 2$, only $c_2(x) = \psi_2$ and $c_3(x) = \frac{1}{2}(\xi_1^2 + \xi_2^2)$ have a nonzero Hessian, constantly equal to a scalar factor of the identity matrix of order $n$ (the corresponding factors are $-0.08$ `cubic` and $0.0016$, respectively).

Our final function, modifying [Lew02, Sec. 7] as in [MS03] to have a smooth inner mapping, is an example of a composite function having a nonconvex positively homogeneous outer function.

**Example 6 (Modified Lewis function)** For $x = (x_1, x_2)^\top$ consider the following function, defined on a partition of $\Re^2$:

$$f(x) := \begin{cases} x_1^2 - x_2 & \text{on} \quad \{(x_1,x_2) \in \Re^2 : x_2 \leq 0\} \\ x_1^2 + x_2 & \text{on} \quad \{(x_1,x_2) \in \Re^2 : 0 < x_2 \leq x_1^2\} \\ 3x_1^2 - x_2 & \text{on} \quad \{(x_1,x_2) \in \Re^2 : 0 < x_1^2 \leq x_2 \leq 4x_1^2\} \\ -5x_1^2 + x_2 & \text{on} \quad \{(x_1,x_2) \in \Re^2 : 4x_1^2 < x_2\}\,. \end{cases}$$

This nonconvex function has the origin as unique critical point. The composite form, $f = h \circ c$, has smooth inner mapping $c : \Re^2 \to \Re^2$ defined by $c_1(x) = x_1^2$ and $c_2(x) = x_2$; and outer function $h : \Re^2 \to \Re$ given by

$$h(C) := \begin{cases} |C_1|^+ - C_2 & \text{on} \quad C \in \{(C_1,C_2) \in \Re^2 : C_2 \leq 0\} \\ |C_1|^+ + C_2 & \text{on} \quad C \in \{(C_1,C_2) \in \Re^2 : 0 < C_2 \leq C_1\} \\ 3|C_1|^+ - C_2 & \text{on} \quad C \in \{(C_1,C_2) \in \Re^2 : 0 < C_1 \leq C_2 \leq C_1\} \\ -5|C_1|^+ + C_2 & \text{on} \quad C \in \{(C_1,C_2) \in \Re^2 : 4C_1 < C_2\}\,. \end{cases}$$

This outer function is posivitely homogeneous, but not convex, so $h \circ c$ is not included in our family.

# 3 Outer function and conceptual model

Even though the composite function $h \circ c$ may be nonconvex, it is always locally Lipschitz continuous and directionally differentiable. Furthermore, as shown in [Sha03, Prop.3.2], the composite function is semismooth [Mif77], and strongly semismooth if the inner mapping is twice continuously differentiable.

Since the outer function is assumed to be positively homogeneous of degree 1, its directional derivative satisfies $h'(0;\cdot) = h(\cdot)$, so $(h \circ c)'(x;d) = h(Dc(x)d)$ for any $x, d \in \Re^n$. As a result, the function $(h \circ c)'(x;\cdot)$ is convex and its convex subdifferential at 0 coincides with the Clarke subdifferential $\partial(h \circ c)(x)$, in turn equal to the regular subdifferential from [RW98, Def. 8.3].

The optimality condition for $\bar{x}$ to be a critical point of (1) is $0 \in \partial(h \circ c)(\bar{x})$. Being real-valued and convex, the outer function is continuous and, using a chain rule [BS00, Ch.3.4.1],

$$\forall x \in \Re^n \quad \partial(h \circ c)(x) = Dc(x)^\top \partial h(C) \quad \text{for } C = c(x)\,. \tag{2}$$

In particular, critical points satisfy the inclusion

$$0 \in Dc(\bar{x})^\top \partial h(\bar{c}) \quad \text{ for } \bar{c} = c(\bar{x})\,.$$

Another important consequence of positive homogeneity is the generalized *Euler formula*, stating that

$$\forall C \in \Re^m \text{ and for any } G \in \partial h(C) \text{ it holds that } h(C) = G^\top C\,. \tag{3}$$

(the result follows from the subgradient inequality $h((\alpha+1)C) \geq h(C) + G^\top((\alpha+1)C - C)$, using that $h((\alpha+1)C) = (\alpha+1)h(C)$ for any $\alpha \geq -1$). This particular structural property will be exploited in the bundle method to define a cutting-planes approximation of the conceptual model.

The conceptual model from [LW08] composes the outer function with the linearization of the inner mapping around a point $x^k$. Specifically, if $D_k = Dc(x^k)$ denotes the Jacobian mapping at $x^k$, then for any $d \in \Re^n$, letting

$$c_k(d) := c(x^k) + D_k d\,,$$

the conceptual model is given by the function

$$h(c_k(d))\,. \tag{4}$$

To make tractable this conceptual model, we use a cutting-planes approximation $\check{h}$ for the outer function and, for each pair $(H^i := h(C^i), G^i \in \partial h(C^i))$, define the plane

$$H^i + G^{i\,\top}(c_k(d) - C^i) = h(c(x^k)) - \Delta_i^k + G^{i\,\top} c_k(d)\,,$$

with

$$\Delta_i^k := h(c(x^k)) - H^i + G^{i\,\top} C^i \geq G^{i\,\top} c(x^k)\,, \tag{5}$$

by the subgradient inequality. The cutting-planes model for the conceptual model has the form

$$\check{h}(c_k(d)) = h(c(x^k)) + \max_{i \in \mathcal{B}} \left\{ -\Delta_i^k + G^{i\,\top} c_k(d) \right\}\,, \tag{6}$$

where the index set $\mathcal{B}$ represents the bundle of information, varying along iterations. By convexity of $h$ the cutting-planes model is always a lower approximation for the conceptual model, as shown in Figure 1 below.

When $h$ is positively homogeneous, the equality (3) implies in (5) that $\Delta_i^k := h(c(x^k))$, so each plane has the form $G^{i\,\top} c_k(d)$, and the bundle only needs to keep the subgradient information $(G^i)$. Accordingly, the composite cutting-planes model can be written as

$$\check{h}(c_k(d)) = \max_{i \in \mathcal{B}} \left\{ G^{i\,\top} c_k(d) \right\}\,. \tag{7}$$

The example in Figure 1 shows that the composite model may not be a lower approximation for $h \circ c$, however, since the conceptual model is a good estimate for $h \circ c$, eventually the composite model is a good approximation for the objective function. This feature is particularly important for nonconvex bundle methods, which need to handle very carefully cutting-planes models (in some cases a tangent line of a nonconvex function can cut-off a section of the graph of the function, leaving out a critical point, for example), [HS10].

Typical situations arising when dealing with nonconvex functions are displayed in Figure 1. Specifically, setting $n = 1$ in Example 4, the function in the figure is defined by adding 4 scalar terms of the form $|\phi_k(x) = 0.5a_k x^2 + b_k x + c_k|$, with $a_1 > 0$ and $a_{2,3,4} < 0$. Both cutting-planes models in the drawing use the same base points $x^i$ in their respective bundles of information. The classical model is defined using the composite data $\{x^i, (h \circ c)(x^i), \gamma^i \in \partial(h \circ c)(x^i)\}$, whereas the composite model uses the outer data $\{G^i \in \partial h(C^i)\,,\ \text{for } C^i = c(x^i)\}$. Finally, the point $x^*$ in the $x$-axis is a critical point found by the composite bundle method given in the next section.

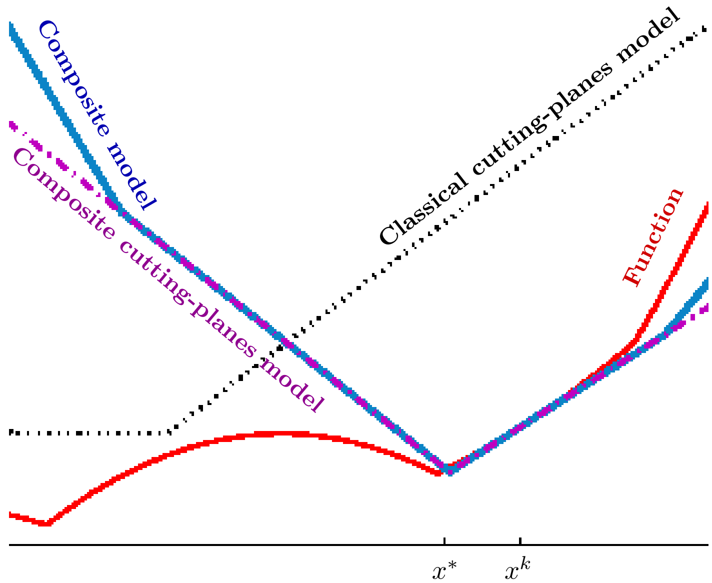



Figure 1: Various models for a nonconvex composite function.

# 4 Composite bundle method

At iteration number $\ell$, the composite model $\check{h}_\ell(c_k(d))$, given by (7) written with bundle set $\mathcal{B} = \mathcal{B}_\ell$, is used to generate a direction $d^\ell$. The inner mapping is linearized at a point $x^k$ with $k = k(\ell)$. It is possible for $k$ to be fixed for several consecutive values of $\ell$. The quadratic programming problem (QP) solved by $d^\ell$ is nothing but the computation of the proximal point of the composite model, with prox-center $x^k$ and a variable metric depending on matrices $M_k^\ell$ detailed below.

The bundle of information $\mathcal{B}_\ell$ is formed from (some of the) past information, corresponding to past directions $d^i$ with $i < \ell$, generated using a prox-center $x^{k(i)}$ and for which a call to the outer black-box ($\mathtt{bb}_h$) with $C^i = c_{k(i)}(d^i)$ gave the value $H^i = h(C^i)$ and the subgradient $G^i \in \partial h(C^i)$.

The variable prox-metric is induced by a positive definite matrix $M_k^\ell$ of order $n$ having the form

$$M_k^\ell = M_k + \mu_\ell \mathrm{I} \text{ with } M_k \text{ a symmetric } n \times n \text{ matrix, } \mu_\ell \geq 0\text{, and I the identity matrix of order } n\,.$$

The corresponding primal and dual metrics in $\Re^n$ are

$$\forall x, \gamma \in \Re^n \quad |x|_{k,\ell}^2 := x^\top M_k^\ell x \qquad \text{and} \qquad \|\gamma\|_{k,\ell}^2 := \gamma^\top (M_k^\ell)^{-1}\gamma\,,$$

respectively. The eigenvalues of $M_k^\ell$ are obtained by adding $\mu_\ell$ to those of $M_k$, while the eigenvectors of $M_k^\ell$ coincide with those of $M_k$. As a result, if for a matrix $M$ the symbol $\lambda(M)$ (respectively $\lambda_{\min\,/\,\max}(M)$) denotes an eigenvalue (resp., minimum/ maximum eigenvalues), then

$$\lambda(M_k^\ell) = \lambda(M_k) + \mu_\ell\,, \quad \lambda_{\min}(M_k^\ell) = \lambda_{\min}(M_k) + \mu_\ell \quad \text{and } \lambda_{\min}((M_k^\ell)^{-1}) = \frac{1}{\lambda_{\max}(M_k) + \mu_\ell}\,.$$

In particular, letting $|\cdot|$ denote the Euclidean norm in $\Re^n$,

$$|x|_{k,\ell}^2 \geq \Big(\lambda_{\min}(M_k) + \mu_\ell\Big)|x|^2 \quad \text{and} \quad \|\gamma\|_{k,\ell}^2 \geq \frac{1}{\lambda_{\max}(M_k) + \mu_\ell}|\gamma|^2\,. \tag{8}$$

Having the elements above, the next direction $d^\ell \in \Re^n$ solves the quadratic programming problem

$$\min_{d \in \Re^n} \check{h}_\ell(c_k(d)) + \frac{1}{2}|d|_{k,\ell}^2\,. \tag{9}$$

Letting $C^\ell = c_{k(\ell)}(d^\ell)$, the corresponding optimality condition with $k = k(\ell)$ is

$$\exists \hat{G}^\ell \in \partial \check{h}_\ell(C^\ell) = conv\{G^i : i \in \mathcal{B}_\ell\} : \begin{cases} M_k^\ell d^\ell + D_k^\top \hat{G}^\ell = 0 \\ \check{h}_\ell(C^\ell) = \hat{G}^{\ell\,\top} C^\ell = \hat{G}^{\ell\,\top} c(x^k) - \|D_k^\top \hat{G}^\ell\|_{k,\ell}^2 . \end{cases} \tag{10}$$

The optimal gradient $\hat{G}^\ell$ is a convex combination only at active $G^i$'s such that $\check{h}_\ell(C^\ell) = G^{i\top} C^\ell = \hat{G}^{\ell\,\top} c(x^k)$.

Without loss of clarity, and depending on the context, we sometimes use the short notations $k$ and $C^\ell$, instead of the longer ones $k(\ell)$ and $c_{k(\ell)}(d^\ell)$.

**Algorithm 1** (Composite Bundle)

Inner and outer black-boxes, $(\texttt{bb}_c)$ and $(\texttt{bb}_h)$ respectively, are available.

**Step 0 (Input and Initialization)**
Select a stopping tolerance $\texttt{tol}_{\texttt{stop}} \geq 0$, two Armijo and Wolfe-like parameters $m_1, m_2 > 0$, and a minimum positive threshold $0 < \mu_{\min} < +\infty$.
Initialize the iteration and serious step counters to $\ell = 0$ and $k = k(\ell) = 0$, respectively.
For a starting point $x^0 \in \Re^n$, call $(\texttt{bb}_c)$ to compute the inner oracle values $C^0 = c(x^0)$ and $D_0 = Dc(x^0)$.
Call $(\texttt{bb}_h)$ to compute the outer oracle values $H^0 = (h \circ c)(x^0)$, $G^0 \in \partial h(C^0)$. Set $\mathcal{B}_0 := \{G^0\}$.
For a symmetric matrix $M_0$ of order $n$, choose a prox-parameter $\mu_0 \geq 0$ ensuring that the matrix $M_0^0 = M_0 + \mu_0 \mathrm{I}$ is positive definite.

**Step 1 (Model Generation and QP Subproblem)** Having the current serious step iterate $x^k$, $c(x^k)$, its associated Jacobian $D_k$, and the bundle $\{G^i\}_{i \in \mathcal{B}_\ell}$, define the composite cutting-planes model function $\check{h}_\ell(c_k(\cdot))$ from (7). A matrix $M_k$ and a scalar $\mu_\ell \geq 0$ such that $M_k^\ell = M_k + \mu_\ell \mathrm{I}$ is positive definite are also available.
Compute $d^\ell$ by solving the quadratic program (9). These calculations include finding optimal simplicial multipliers $\alpha^\ell$ such that in (10)

$$\hat{G}^\ell = \sum_{i \in \mathcal{B}_\ell} \alpha_i^\ell G^i .$$

Define the aggregate error

$$\hat{e}_\ell = (h \circ c)(x^k) - \hat{G}^{\ell\,\top} c(x^k) ,$$

(note that $\hat{e}_\ell \geq 0$ by (5) and (3)), and compute the predicted decrease

$$\delta_\ell = (h \circ c)(x^k) - \check{h}_\ell(c_k(d^\ell)) - \frac{1}{2}|d^\ell|_{k,\ell}^2 = \hat{e}_\ell + \frac{1}{2}\|D_k^\top \hat{G}^\ell\|_{k,\ell}^2 , \tag{11}$$

where the last equality follows from (10).

**Step 2 (Stopping test)**
If $\max(\hat{e}_\ell, \|D_k^\top \hat{G}^\ell\|_{k,\ell}^2) \leq \texttt{tol}_{\texttt{stop}}$, stop.
Otherwise, call $(\texttt{bb}_h)$ to obtain $h(c_k(d^\ell))$ and $G^\ell \in \partial h(c_k(d^\ell))$.

**Step 3 (Serious/backtrack/null step test)**
Check the descent condition

$$h(c_k(d^\ell)) \leq (h \circ c)(x^k) - m_1 \delta_\ell . \tag{12}$$

If (12) does not hold, declare a null step: take $\mu_{\ell+1} \geq \mu_\ell$, $M_k^{\ell+1} = M_k + \mu_{\ell+1}\mathrm{I}$, and go to Step 4.
If (12) is true, call the inner oracle $(\texttt{bb}_c)$ to compute $c(x^k + d^\ell)$. Call the outer oracle $(\texttt{bb}_h)$ to compute $\Gamma^\ell \in \partial h(c(x^k + d^\ell))$, and check the condition below:

$$\Gamma^{\ell\,\top}[c_k(d^\ell) - c(x^k + d^\ell)] \geq -m_2 \delta_\ell . \tag{13}$$

If (13) holds, declare a serious step, and go to Step 4.
Otherwise, if (13) does not hold, declare a backtracking step: take $\mu_{\ell+1} \geq \mu_{\min} + \mu_\ell$, $M_k^{\ell+1} = M_k + \mu_{\ell+1}\mathrm{I}$, $\mathcal{B}_{\ell+1} = \mathcal{B}_\ell$, set $k(\ell+1) = k(\ell)$, increase $\ell$ by 1, and loop to Step 1.

**Step 4 (Bundle update and management)**
If needed, compress $\mathcal{B}_\ell$ either by keeping all strongly active elements ($i \in \mathcal{B}_\ell : \alpha_i^\ell > 0$), or by replacing some strongly active elements by the aggregate gradient $\hat{G}^\ell$. Define $\mathcal{B}_{\ell+1}$ by adding to the possibly

compressed set the new outer gradient $G^\ell$.
If the step was declared null, set $k(\ell+1) = k(\ell)$, increase $\ell$ by 1, and loop to Step 1.
If the step was declared serious, set $k(\ell+1) = k+1$, $x^{k+1} = x^k + d^\ell$, call ($\texttt{bb}_c$) to compute $D_{k+1} = Dc(x^{k+1})$, compute a new symmetric matrix $M_{k+1}$ and a prox-parameter $\mu_{\ell+1} \geq 0$ such that $M_{k+1}^{\ell+1} = M_{k+1} + \mu_{\ell+1}\mathrm{I}$ is positive definite. Increase $k$ and $\ell$ by 1, and loop to Step 1. □

In Step 3, the decision to declare a null-steps is different from standard bundle methods, and exploits the structure knowledge in the composite case. Namely, while condition (12) checks if there is descent for the outer function, (13) checks adequacy between the inner mapping and its conceptual model.

The prox-parameter increases at backtracking steps, and can be increased at null steps or decreased at serious steps. In this context, the wording "backtrack" may sound odd, since it corresponds to choosing a larger prox-parameter. The explanation comes from noticing that, due to (10), the value $1/\mu_\ell$ can be seen as a stepsize. In this sense, when the algorithm detects the need of backtracking, increasing the prox-parameter results indeed in a smaller stepsize. Since if the matrix $M_k$ is positive definite, we may choose a null prox-parameter, in the backtracking step we force positivity by means of the positive threshold $\mu_{\min}$, set at the Initialization step.

In order to ensure that matrices $M_k^\ell$ are always positive definite, the prox-parameter update can be done as follows:

$$\mu_\ell = \begin{cases} 0 & \text{if } \lambda_{\min}(M_{k(\ell)}) \geq \mu_{\min}\,, \\ -\lambda_{\min}(M_{k(\ell)}) + \mu_{\min} & \text{otherwise.} \end{cases} \tag{14}$$

We mention that there is a difference between the Composite Algorithm 1 and usual bundle methods, in terms of oracle calls. A standard bundle method makes one call to both the inner and outer oracles per iteration, independent of getting a serious or a null step. Instead, Step 2 in Algorithm 1 makes one call to the outer oracle ($\texttt{bb}_h$) at all iterations, needing additional calls to both oracles ($\texttt{bb}_{h/c}$) at Step 3, for deciding between serious or backtracking steps. In this case, the additional subgradient $\Gamma^\ell \in \partial h(c(x^k + d^\ell))$ may enter the bundle.

In Step 4 we see that the bundle is formed only by outer gradients, corresponding either to $G^i \in \partial h(C^i)$ or to some past aggregate gradient $\hat{G}^i$. By convexity of $h$, the cutting-planes model is always a lower approximation to $h$:

$$\text{for all } \ell \geq 1 \text{ and } C \in \Re^m \qquad \check{h}_\ell(C) \leq h(C)\,. \tag{15}$$

Moreover, since $\hat{G}^\ell \in \partial h(C^\ell)$ by (10), after some algebra involving $\delta_\ell$ and $\hat{e}_\ell$, we see that

$$h(c(x^k)) - \check{h}_\ell(C^\ell) - \|D_k^\top \hat{G}^\ell\|_{k,\ell}^2 = \delta_\ell - \frac{1}{2}\|D_k^\top \hat{G}^\ell\|_{k,\ell}^2 = \hat{e}_\ell\,,$$

and, hence,

$$\forall \ell \text{ and } C \in \Re^m \qquad h(C) \geq (h \circ c)(x^k) + \hat{G}^{\ell\,\top}(C - c(x^k)) - \hat{e}_\ell\,. \tag{16}$$

**Remark 1 (The trivial composite case)** *To see how a classical bundle method compares to Algorithm 1, consider a convex nonsmooth function $f$, and the, somewhat artificial, outer function $h \equiv f$ and inner mapping $c(x) = x$. With respect to our composite structure, $h$ is no longer positive definite, $Dc = \mathrm{I}$, $c_k(d) = x^k + d = c(x^k + d)$, and, instead of* (7)*, the cutting-planes model has the form* (6)*. Since the inner mapping is affine, there is no second order to exploit, and the matrices $M_k \equiv 0$ in Algorithm 1. The QP optimality condition in* (10) *becomes*

$$\exists \hat{G}^\ell \in \partial \check{h}_\ell(x^\ell) = conv\{G^i : i \in \mathcal{B}_\ell\} : \begin{cases} \mu_\ell d^\ell + \hat{G}^\ell = 0 \\ \check{h}_\ell(x^\ell) = h(x^k) - \hat{\Delta}_\ell^k + \hat{G}^{\ell\,\top} x^\ell\,, \end{cases}$$

*for $\hat{\Delta}_\ell^k := \sum_{i \in \mathcal{B}_\ell} \alpha_i^\ell \Delta_i^k$. The aggregate error is defined by $\hat{e}_\ell := \hat{\Delta}_\ell^k - \hat{G}^{\ell\,\top} x^k$, consistent with the error definition in Step 1, because when $h$ is positively homogenous, $\Delta_i^k = h(x^k)$, by* (3)*. As a result, the expression for the predicted decrease in* (11) *remains true. Finally, in Step 3, because $c_k(d^\ell) = x^k + d^\ell = c(x^k + d^\ell)$, there is no backtracking step, because the inequality in* (13) *trivially holds:*

$$0 = \Gamma^{\ell\,\top}(c_k(d^\ell) - c(x^k + d^\ell)) \leq -m_2 \delta_\ell\,.$$

*From the above it follows that Algorithm 1 with $M_k \equiv 0$ is applicable for the trivial composite case, becoming a classical bundle method for convex functions without any particular structure.*

## 5 Convergence results

Since matrices $M_k^\ell$ are always positive definite, they induce a norm and (8) holds. Together with (11) and (10), this means that the relations

$$\delta_\ell \geq \begin{cases} \max\Big(\frac{1}{2}\|D_k^\top \hat{G}^\ell\|_{k,\ell}^2, \hat{e}_\ell\Big) & \geq & \dfrac{1}{2(\lambda_{\max}(M_k)+\mu_\ell)}|D_k^\top \hat{G}^\ell|^2 & \text{(a)} \\ \max\Big(\frac{1}{2}|d^\ell|_{k,\ell}^2, \hat{e}_\ell\Big) & \geq & \dfrac{1}{2}\Big(\lambda_{\min}(M_k)+\mu_\ell\Big)|d^\ell|^2 & \text{(b)} \end{cases} \tag{17}$$

are always satisfied.

We start our convergence analysis showing that there can only be a finite number of consecutive backtracking steps.

**Proposition 1 (Finite backtracking loop)** *Suppose the inner mapping is $C^2$. If after some iteration $\hat{\ell}$, Algorithm 1 makes a last serious step and thereafter generates only null and backtracking steps, the following holds:*

*(i) If the sequence $\{d^\ell\}_{\ell\geq\hat{\ell}}$ is bounded, then there exists $\mu_{noBT} > 0$ such that (13) holds for any $\mu_\ell > \mu_{noBT}$.*

*(ii) There cannot be infinitely many consecutive backtracking steps.*

**Proof:** For convenience, let $\hat{k} = k(\hat{\ell})$, $\hat{x} = x^{\hat{k}}(= x^{k(\ell)}$ for all $\ell \geq \hat{\ell})$, and $\hat{M} = M_{\hat{k}}$ denote, respectively, the $k$-iteration index, prox-center and matrix corresponding to the last serious step.

Since $c$ is a $C^2$-mapping, a mean-value theorem applies to each component $c_j$, $j = 1, \ldots, m$:

$$c_j(\hat{x}+d^\ell) - c_j(\hat{x}) - \nabla c_j(\hat{x})^\top d^\ell = \frac{1}{2}\nabla^2 c_j(\xi_j)(d^\ell, d^\ell) \quad \text{for some } \xi_j \in [\hat{x}, \hat{x}+d^\ell]\,.$$

Recall that $|\cdot|$ is the Euclidean matrix norm. By assumption, $\{\hat{x}+d^\ell\}$ is bounded so $|\Gamma^\ell| \leq L$ for all $\ell$ and some constant $L$. Similarly, boundedness of $\{d^\ell\}$ implies that $|\nabla^2 c_j(\xi_j)| \leq D$ for all $j = 1, \ldots, m$, for some constant $D$. By Cauchy-Schwarz inequality,

$$\Gamma^{\ell\,\top}[c(\hat{x}+d^\ell) - c(\hat{x}) - D_{\hat{k}} d^\ell] \leq \frac{\sqrt{m}}{2} LD|d^\ell|^2\,.$$

Using the rightmost inequality in (17)(b), we obtain

$$\delta_\ell \geq \frac{1}{2}|d^\ell|_{\hat{k},\ell}^2 \geq \frac{1}{2}(\lambda_{\min}(\hat{M})+\mu_\ell)|d^\ell|^2\,.$$

In the notation used in this proof, $c(x^k+d^\ell) = c(\hat{x}+d^\ell)$ and $c_k(d^\ell) = c(\hat{x}) + D_{\hat{k}} d^\ell$. Suppose that (13) is not satisfied and multiply by $-1$ the corresponding inequality:

$$m_2\delta_\ell < -\Gamma^{\ell\,\top}[c_k(d^\ell) - c(x^k+d^\ell)] = \Gamma^{\ell\,\top}[c(\hat{x}+d^\ell) - c(\hat{x}) - D_{\hat{k}} d^\ell]\,.$$

Using the bounds above we see that

$$\frac{1}{2}m_2(\lambda_{\min}(\hat{M})+\mu_\ell)|d^\ell|^2 \leq m_2\delta_\ell < \Gamma^{\ell\,\top}[c(\hat{x}+d^\ell) - c(\hat{x}) - D_{\hat{k}} d^\ell] \leq \frac{\sqrt{m}}{2} LD|d^\ell|^2\,.$$

Therefore, nonsatisfaction of (13) implies $\mu_\ell < \sqrt{m}LD/m_2 - \lambda_{\min}(\hat{M})$. Item (i) follows, by taking $\mu_{\rm noBT} \geq \sqrt{m}LD/m_2 - \lambda_{\min}(M_k)$.
The claim in item (ii) is shown by contradiction, assuming that (13) does not hold, with $\ell \to \infty$ due to backtracking. An infinite backtracking loop drives $\mu_\ell$ to infinity, as well as the minimum eigenvalue of the matrix $M_{\hat{k}}^\ell$, because $M_{\hat{k}}^\ell = \hat{M} + \mu_\ell \mathrm{I}$. Since there are only backtracking steps, the bundle does not change, and the composite model $\check{h}_\ell(c_{\hat{k}}(\cdot))$ is a fixed function, say $\varphi$. Hence, by [HUL93, Prop. XV.4.1.5], the minimand in (9) converges to $\varphi(d)$ for all $d \in \Re^n$ and $d^\ell \to 0$ as $\ell \to \infty$. Therefore, $\hat{x} + d^\ell \to \hat{x}$ with $\varphi(d^\ell) + \frac{1}{2}|d^\ell|_{\hat{k},\ell}^2 \to \varphi(0)$. In particular, this means that the sequence $\{d^\ell\}$ is bounded, and the contradiction follows from item (i). $\square$

As a consequence, if the algorithm loops forever, it generates either an infinite sequence of serious steps, or a finite one, followed by infinitely many null steps (possibly nonconsecutive, due to backtracking). Both situations are considered below, adapting classical developments for bundle methods [HUL93, Ch.XV.3, XV.4] to the composite setting.

We start with the case of infinitely many serious steps.

**Lemma 1 (Infinitely many serious steps)** *Suppose Algorithm 1 generates an infinite sequence $\{x^k\}$ of serious steps. Denote by $\ell_k$ an iteration index yielding a serious step: $x^{k+1} = x^k + d^{\ell_k}$. The following holds:*

*(i) If $0 < m_2 < m_1$ then, as $k \to \infty$, either $(h \circ c)(x^k) \to -\infty$ or $\delta_{\ell_k} \to 0$.*

*(ii) Suppose $\delta_{\ell_k} \to 0$. Then $\lim \hat{e}_{\ell_k} = 0$ and,*

$$\text{if the series } \sum_k \frac{1}{\mu_{\ell_k} + \lambda_{\max}(M_k)} \text{ is divergent,} \tag{18}$$

*then $\liminf \|D_k^\top \hat{G}^{\ell_k}\|_{k,\ell_k} = 0$.*

*(iii) If, in addition, the sequence $\{x^k\}$ is bounded, then it has at least one accumulation point that is critical for (1).*

*(iv) If, instead of (18), the stronger condition*

$$\text{the sequence } \{\mu_{\ell_k} + \lambda_{\max}(M_k)\} \text{ is bounded above,} \tag{18'}$$

*then all accumulation points are critical for (1).*

**Proof:** By (13), since $\Gamma^{\ell_k} \in \partial h(c(x^k + d^{\ell_k}))$, we have that

$$h(c(x^k) + D_k d^{\ell_k}) \geq h(c(x^k + d^{\ell_k})) - m_2\delta_{\ell_k} = (h \circ c)(x^{k+1}) - m_2\delta_{\ell_k} \,. \tag{19}$$

Together with satisfaction of (12), this means that $(h \circ c)(x^{k+1}) \leq (h \circ c)(x^k) - (m_1 - m_2)\delta_{\ell_k}$. The telescopic sum yields that either $(h \circ c)(x^k) \searrow -\infty$ or $0 \leq \delta_{\ell_k} \to 0$, because $m_1 - m_2 > 0$.
The first assertion in item (ii) follows from (11). The rightmost inequality in (17)(a), together with our assumption (18) gives the second result.
To show item (iii), consider a $k$-subsequence such that $\|D_k^\top \hat{G}^{\ell_k}\|_{k,\ell_k} \to 0$ and extract a further subsequence of serious steps with accumulation point $x^{acc}$. By boundedness of $\{x^k\}$ and local Lipschitzianity of $h$, there is an associated subsequence $\{\hat{G}^\ell\}$ with accumulation point $\hat{G}^{acc}$. Passing to the limit in (16) and using that $\hat{e}_{\ell_k} \to 0$, we obtain in the limit that $\hat{G}^{acc} \in \partial h(C^{acc})$ for $C^{acc} = c(x^{acc})$. By smoothness of $c$, $D_k \to D_{acc} = Dc(x^{acc})$, and by (2), $D_{acc}^\top \hat{G}^{acc} \in \partial(h \circ c)(x^{acc})$. The result follows from item (ii).
Item (iv) is similar to item (iii), noticing that if $L > 0$ denotes an upper bound for $\{\mu_{\ell_k} + \lambda_{\max}(M_k)\}$, then the relation $\frac{1}{2(\mu_{\ell_k} + \lambda_{\max}(M_k))} \geq 1/2L$ in (17)(b) gives that $|D_k^\top \hat{G}^{\ell_k}| \to 0$ as $k \to \infty$ (for the whole sequence). $\square$

The remaining case refers to an infinite null-step loop and makes use of the *aggregate linearization*

$$\check{h}_\ell^{\text{agg}}(c_k(d)) = \check{h}_\ell(c_k(d^\ell)) + \hat{G}^{\ell\,\top} D_k(d - d^\ell)\,,$$

the associated strongly convex function

$$\mathcal{H}_\ell(d) = \check{h}_\ell^{\text{agg}}(c_k(d)) + \frac{1}{2}|d|^2_{k,\ell}\,, \tag{20}$$

and the result [HUL93, Lem. XV.4.3.3]:

$$\mathcal{H}_{\ell-1}(d) = \mathcal{H}_{\ell-1}(d^{\ell-1}) + \frac{1}{2}|d - d^{\ell-1}|^2_{k,\ell-1}\,. \tag{21}$$

Finally, and similar to [CL93, Sec. 4], note that Step 4 in Algorithm 1 updates the bundle of information in a way ensuring that not only

$$\text{if iteration } \ell - 1 \text{ was declared a null step, then} \qquad \check{h}_{\ell-1}^{\text{agg}}(c_k(d)) \leq \check{h}_\ell(c_k(d))\,, \tag{22}$$

but also

$$\text{if iteration } \ell - 1 \text{ was declared a null step, then} \qquad \check{h}_\ell(c_k(d)) \geq G^{\ell-1\,\top} c_k(d) \tag{23}$$

for all $d \in \Re^n$.

**Lemma 2 (Finitely many serious steps)** *Suppose that, after some iteration $\hat\ell$, Algorithm 1 makes a last serious step $\hat x = \hat x^{k(\ell)}$ and thereafter generates an infinite number of null steps, possibly nonconsecutive, due to intermediate backtracking steps. The following holds:*

*(i) The sequence $\{d^\ell\}_{\ell>\hat\ell}$ is bounded and there is an iteration $\ell' > \hat\ell$ such that only null steps are done for all $\ell \geq \ell'$.*

*(ii) If $m_1 < 1$, then $\delta_\ell \to 0$ and $\hat e_\ell \to 0$.*

*(iii) If, in addition, for the (fixed) matrix $\hat M = M_{k(\hat\ell)}$.*

$$\text{the series} \sum_{\ell\geq\ell'} \frac{\mu_{\ell-1} + \lambda_{\min}(\hat M)}{(\mu_\ell + \lambda_{\max}(\hat M))^2} \text{ is divergent,} \tag{24}$$

*then $\liminf |Dc(\hat x)^\top \hat G^\ell| = 0$, $\hat x + d^\ell \to \hat x$ for some $\ell$-subsequence, and $\hat x$ is critical for* (1).

**Proof:** For convenience, let $\hat k = k(\hat\ell)$, $\hat x = x^{\hat k} (= x^{k(\ell)}$ for all $\ell \geq \hat\ell)$, and $\hat M = M_{\hat k}$ denote, respectively, the $k$-iteration index, prox-center and matrix corresponding to the last serious step.

Consider $\ell > \hat\ell$ and recall that, since $M^\ell_{\hat k} = \hat M + \mu_\ell \mathrm{I}$ and $\{\mu_\ell\}$ is nondecreasing at null and backtracking steps,

$$\frac{1}{2}|d|^2_{\hat k,\ell-1} \leq \frac{1}{2}|d|^2_{\hat k,\ell}\,.$$

The sum of this inequality and (22), together with (20) written with $\ell$ replaced $\ell-1$, results in the relation

$$\mathcal{H}_{\ell-1}(d) \leq \check h_\ell(c_{\hat k}(d)) + \frac{1}{2}|d|^2_{\hat k,\ell}\,.$$

In particular, for $d = d^\ell$, we obtain that $\mathcal{H}_{\ell-1}(d^\ell) \leq \mathcal{H}_\ell(d^\ell)$ from (20), because $\check h_\ell(c_{\hat k}(d^\ell)) = \check h^{\mathrm{agg}}_\ell(c_{\hat k}(d^\ell))$. Together with (21), written at $d = d^\ell$, we see that

$$\mathcal{H}_{\ell-1}(d^{\ell-1}) \leq \mathcal{H}_{\ell-1}(d^{\ell-1}) + \frac{1}{2}|d^\ell - d^{\ell-1}|^2_{\hat k,\ell-1} = \mathcal{H}_{\ell-1}(d^\ell) \leq \mathcal{H}_\ell(d^\ell)\,. \tag{25}$$

By the definitions of $\check h^{\mathrm{agg}}_\ell$ and $\mathcal{H}_\ell$, the optimal value in (9) equals $\mathcal{H}_\ell(d^\ell)$. Hence, (25) implies that the sequence of optimal values in (9) is strictly increasing, with $\mathcal{H}_\ell(d^\ell) \leq \check h_\ell(c_k(0)) = \check h_\ell(c(\hat x))$. Since, by (15), $\check h_\ell \leq h$, then

$$\mathcal{H}_\ell(d^\ell) \leq (h\circ c)(\hat x)\,. \tag{26}$$

So $\{\mathcal{H}_\ell(d^\ell)\} \uparrow \mathcal{H}_\infty$ for some $\mathcal{H}_\infty \leq (h\circ c)(\hat x)$, with $|d^\ell - d^{\ell-1}|^2_{\hat k,\ell-1} \to 0$ as $\ell\to\infty$, by (25). But $\mu_\ell \geq \mu_{\hat\ell}$ (at null and backtracking steps prox-parameters are nondecreasing), so the left relation in (8) implies that

$$d^\ell - d^{\ell-1} \to 0\,. \tag{27}$$

Using (20) with $d = 0$, we see that $\mathcal{H}_\ell(0) = \check h^{\mathrm{agg}}_\ell(c_{\hat k}(0))$. Together with the definition of $\check h^{\mathrm{agg}}_\ell$, the left inequality in the second line in (10), and the definition of $c_k(d^\ell)$, this implies that that $\mathcal{H}_\ell(0) = \check h_\ell(c_{\hat k}(d^\ell)) - \hat G^{\ell\,\top} D_{\hat k} d^\ell = \hat G^{\ell\,\top} c(\hat x)$. Since $\hat G^\ell \in conv\{G^i : i \in \mathcal{B}_\ell\}$, using (5) and (3) we obtain that $\mathcal{H}_\ell(0) \leq (h\circ c)(\hat x)$. Therefore, writing (21) with $\ell-1$ replaced by $\ell$ at $d = 0$ yields the relations

$$\frac{1}{2}|d^\ell|^2_{\hat k,\ell} = \mathcal{H}_\ell(0) - \mathcal{H}_\ell(d^\ell) \leq (h\circ c)(\hat x) - \mathcal{H}_{\hat\ell+1}(d^{\hat\ell+1})\,,$$

because the sequence $\{\mathcal{H}_\ell(d^\ell)\}$ is increasing, by (25), since $\ell > \hat\ell$. Using once more the left relation in (8) we conclude that the sequence $\{d^\ell\}$ is bounded, and item (i) follows from Proposition 1(i).
To show item (ii), consider iteration indices $\ell-1, \ell \geq \ell'$, giving two consecutive null steps, and set $C^\ell = c(\hat x) + Dc(\hat x)d^\ell$ and $C^{\ell-1} = c(\hat x) + Dc(\hat x)d^{\ell-1}$. Since $\delta_\ell \geq 0$, we substract the inequality $\delta_\ell \leq (h\circ c)(\hat x) - \check h_\ell(C^\ell)$, obtained from (11), from nonsatisfaction of (12), both with $x^k = \hat x$, to see that

$$0 \leq (1-m_1)\delta_\ell \leq h(C^\ell) - \check h_\ell(C^\ell)\,. \tag{28}$$

By (3),

$$h(C^{\ell-1}) = G^{\ell-1\,\top} C^{\ell-1}\,,$$

and by (23),

$$\check{h}_\ell(c_{\hat{k}}(d^\ell)) = \check{h}_\ell(C^\ell) \geq G^{\ell-1\,\top} C^\ell\,.$$

Since $\{d^\ell\}$ is bounded, any Lipschitz constant $L$ for $h$ gives an upper bound for $\{|G^\ell|\}$; as a result,

$$\begin{array}{rcl} h(C^\ell) - \check{h}_\ell(C^\ell) & = & h(C^\ell) - h(C^{\ell-1}) + h(C^{\ell-1}) - \check{h}_\ell(C^\ell) \\ & \leq & L|C^\ell - C^{\ell-1}| + G^{\ell-1\,\top}(C^{\ell-1} - C^\ell) \\ & \leq & 2L|Dc(\hat{x})||d^\ell - d^{\ell-1}|\,. \end{array} \tag{29}$$

From (28) and (27) it follows that $\delta_\ell \to 0$, and by the right hand side expression in (11), $\hat{e}_\ell \to 0$, as stated.
Finally, to see item (iii), the left hand side expression in (11) of $\delta_\ell$ and the definitions of $\check{h}_\ell^{\rm agg}$ and $\mathcal{H}_\ell$ give the identity $\delta_\ell = (h \circ c)(\hat{x}) - \mathcal{H}_\ell(d^\ell)$. Therefore, by the right inequality in (25), (22) with $d = d^\ell$, and the left relation in (8),

$$\begin{array}{rcl} \delta_{\ell-1} & \geq & \delta_\ell + \dfrac{1}{2}|d^\ell - d^{\ell-1}|^2_{\hat{k},\ell-1} \\ & \geq & \delta_\ell + \dfrac{\lambda_{\min}(\hat{M}) + \mu_{\ell-1}}{2}|d^\ell - d^{\ell-1}|^2\,. \end{array}$$

From (29) and (28), we obtain that $\delta_\ell \leq \frac{2L|Dc(\hat{x})|}{1-m_1}|d^\ell - d^{\ell-1}|$, so

$$\delta_{\ell-1} - \delta_\ell \geq \frac{\lambda_{\min}(\hat{M}) + \mu_{\ell-1}}{2}\Big(\frac{1-m_1}{2L|Dc(\hat{x})|}\Big)^2 \delta_\ell^2\,.$$

Letting $K := (1-m_1)^2/\Big(8|Dc(\hat{x})|^2 L^2\Big)$, and summing over $\ell > \hat{\ell}$,

$$K \sum_{\ell > \hat{\ell}} (\lambda_{\min}(\hat{M}) + \mu_{\ell-1})\delta_\ell^2 \leq \delta_{\hat{\ell}} < +\infty\,.$$

Furthermore, using (17)(a), the series

$$\sum_{\ell > \hat{\ell}} \frac{\mu_{\ell-1} + \lambda_{\min}(\hat{M})}{(\mu_\ell + \lambda_{\max}(\hat{M}))^2}|Dc(\hat{x})^\top \hat{G}^\ell|^4$$

converges too. With our assumption (24), this implies that $\liminf |Dc(\hat{x})^\top \hat{G}^\ell|^4 = 0$. Consider indices $\ell$ in a corresponding convergent subsequence of $\{d^\ell\}$. Since from the expression for $d^\ell$ in (10), $|Dc(\hat{x})^\top \hat{G}^\ell|^2 = |M_k^\ell d^\ell|^2 \geq (\lambda_{\min}(\hat{M}) + \mu_{\hat{\ell}})^2 |d^\ell|^2$, we see that the subsequence fo $\{d^\ell\}$ converges to zero and, hence, $\hat{x} + d^\ell \to \hat{x}$ on this subsequence. Finally, by boundedness of $\{C^\ell = c(\hat{x}) + Dc(\hat{x})d^\ell\}$, all outer subgradients are bounded and, hence, the subsequence $\{\hat{G}^\ell\}$ has some accumulation point $\hat{G}^{acc}$ such that $Dc(\hat{x})^\top \hat{G}^{acc} = 0$. The result follows from passing to the limit in (16) to give $\hat{G}^{acc} \in \partial h(c(\hat{x}))$, and using the chain rule (2). $\square$

**Remark 2 (The trivial composite case, suite and end.)** *In the setting of Remark 1, when h is not positively homogeneous but merely convex, and c is the identity, the cutting-planes model has the form* (6) *and* (23) *states that*

$$\textit{if iteration } \ell - 1 \textit{ was declared a null step, then} \qquad \check{h}_\ell(c_k(d)) \geq h(x^k) - \Delta^k_{\ell-1} + G^{\ell-1\,\top} c_k(d)\,,$$

*which is consistent with the fact that $h(x^k) = \Delta^k_{\ell-1}$ when h is positively homogenous, by* (5) *and* (3).
*In fact, when $M_k \equiv 0$ for all k, as considered in the trivial structure, Lemmas 1 and 2 boil down to [HUL93, pp.309 and 311, vol.II, Thms.XV.3.2.2 and XV.3.2.4].*

Putting together Proposition 1 and Lemmas 1 and 2, we can show convergence for objective functions that are *inf-compact*, i.e., functions having some level set that is nonempty and compact (sometimes also referred to as *level-bounded* functions). In this case, by lower semicontinuity, $h \circ c$ always attains its minimum and the sequence of serious steps is bounded.

**Theorem 1 (Synthesis)** *Consider solving problem* (1) *with Algorithm 1 and suppose that in* (1) *the objective function* $h \circ c$ *is inf-compact. If* $0 < m_2 < m_1 < 1$, $\texttt{tol}_{\texttt{stop}} = 0$, *with both* (18) *and* (24) *being satisfied, the following holds.*

*(i) Either the sequence of serious steps is infinite and bounded, and at least one of its accumulation points is critical for* (1).

*(ii) Or the last serious step* $\hat{x}$ *is critical for* (1), *with* $\{\hat{x} + d^\ell\} \to \hat{x}$ *for some* $\ell$*-subsequence.*

*If, instead of* (18), *the stronger condition* (18') *holds, item* (*i*) *can be replaced by*

*(i') Either the sequence of serious steps is infinite and bounded, and all its accumulation points are critical for* (1). □

Our conditions (18) and (24), on the variable prox-metric, are fairly general and not difficult to enforce. For (18) to hold, it is enough to take matrices $M_k^\ell$ that are uniformly bounded from above at serious steps (when $\ell = \ell_k$). As for null steps, choosing $\mu_{\ell+1} \in [\mu_\ell, \mu_{\max}]$ for some finite bound $\mu_{\max} > -\lambda_{\min}(\hat{M})$, rule (14) ensures satisfaction of (24). Depending on the particular problem, the more general condition (24) may help in preventing bad choices (too small) for the bound $\mu_{\max}$.

We finish our analysis by supposing the stopping tolerance is positive. In this case, if (18') and (24) hold, by Lemmas 1 and 2, the nominal decrease goes to 0. Then, by (11), both $\hat{e}_\ell$ and $\hat{G}^\ell$ go to 0, and eventually the stopping test will be triggered. For such an iteration index, say $\ell_{\text{best}}$, (16) gives the following approximate optimality condition for the last serious step $x^{\text{best}} = x^{k(\ell\text{best})}$:

$$\begin{aligned}
\forall x \in \Re^n \qquad (h \circ c)(x) &\geq (h \circ c)(x^{\text{best}}) + \hat{G}^{\ell\text{best}\,\top}\Big(c(x) - c(x^{\text{best}})\Big) - \hat{e}_{\ell\text{best}} \\
&\geq (h \circ c)(x^{\text{best}}) - \hat{G}^{\ell\text{best}\,\top}\Big(Dc(x^{\text{best}})(x - x^{\text{best}}) + o(|x - x^{\text{best}}|)\Big) - \hat{e}_{\ell\text{best}} \\
&\geq (h \circ c)(x^{\text{best}}) - \texttt{tol}_{\texttt{stop}}|x - x^{\text{best}}| - \texttt{tol}_{\texttt{stop}} + o(|x - x^{\text{best}}|)\,.
\end{aligned}$$

# 6 Numerical experience

In order to assess from a practical point of view the Composite Bundle method, we coded Algorithm 1 in MATLAB and ran it on several collections of functions described in Sec. 2, using a computer with one 3GHz processor and 1.49GB RAM.

## 6.1 Solvers in the benchmark

We compared the performance of the Composite Bundle method with the MATLAB HANSO package, implementing a "Hybrid Algorithm for NSO", and downloadable from `http://cs.nyu.edu/overton/software/index.html`. As explained in [LO08], for nonsmooth optimization problems, BFGS may fail theoretically. However, the HANSO package can provide good benchmarks for the purpose of comparison, helping to shed some light on important NSO issues, as shown in our results below.

The package is organized in two phases:

– A first phase runs a BFGS method for smooth unconstrained optimization, with a linesearch capable of handling kinks explained in [LO08], and with multiple starting points. If the termination test is not satisfied at the best point found by BFGS, HANSO continues to the next phase.

– The second phase executes up to three runs of the Gradient Sampling method in [BLO02], starting from the lowest point found by BFGS, and with decreasing sampling radii. As initial information, the Gradient Sampling uses *BFGS final bundle*, corresponding to the last $\min[100, 2n, n+10]$ generated points and their `(bb)` information. For locally Lipschitz functions, the method converges to Clarke critical points in a probabilistic sense; [BLO02].

We also created another hybrid algorithm, the *Hybrid Composite Bundle*, that after the BFGS phase switches to the Composite Bundle method, starting like HANSO from BFGS's final bundle.

Therefore, the benchmark considers the four solvers below:

– CBUN, the Composite Bundle method,

– BFGS, the first phase in the HANSO package,

– HANSO, the hybrid variant combining BFGS and Gradient Sampling methods; and

– HYCB, the hybrid variant combining BFGS and CBUN.

## 6.2 Parameters for the different solvers

Letting $n$ be the problem dimension, the maximum number of iterations and calls to the oracle were set to $\texttt{maxit} = 150\min(n, 20)$ and $\texttt{maxsim} = 300\min(n, 20)$, respectively. We set the stopping tolerance $\texttt{tol}_{\texttt{stop}} = 10^{-5}$ if $n < 50$ and multiply it by $\sqrt{n}$ when $n \geq 50$.

### 6.2.1 Parameters for CBun and HyCB

The two Armijo and Wolfe-like parameters are $m_1 = 0.9$ and $m_2 = 0.55$, respectively. The minimum and maximum positive thresholds are $\mu_{\min} = 10^{-6}$ and $\mu_{\max} = 10^8$. In all of our runs, only strongly active bundle elements are kept at each iteration, but, when there are more than $|\mathcal{B}\max| = 50$ strongly active elements, the bundle is compressed.

If no second-order information is available for the inner mapping, in the variable prox-metric we take $M_k = 0$. Otherwise, we use Newton-type matrices

$$M_k = \sum_{j=1}^{m} \hat{G}_j^{\ell_k} \nabla^2 c_j(x^k)\,, \tag{30}$$

exploiting sparsity patterns, if they exist. To update the prox-parameter, we use an estimate for $\lambda_{\min}(M_k)$, obtained by a modified Cholesky factorization.

The prox-parameter was started at

$$\mu_0 = \frac{5|\gamma(x^0)|^2 10^{-\texttt{fact}}}{1 + |(h \circ c)(x^0)|}\,.$$

The parameter $\texttt{fact}$ is 0 if the function is convex, and 3 otherwise (a large value for this parameter constrains the search of the next iterate in a region close to $x^0$, which makes sense for a nonconvex function). At later iterations, every time a serious step is declared, the update is done as follows:

$$\mu_{\ell+1} = \min(\mu, \mu_{\max}) \text{ for } \mu = \begin{cases} \max(\mu_{\min}, \mu_\ell, -1.01\lambda_{\min}(M_k)) & \text{if } \lambda_{\min}(M_k) < 0 \\ \max(\mu_{\min}, \mu_\ell^{\texttt{qN}}) & \text{if } \lambda_{\min}(M_k) = 0 \\ \max(0, \lambda_{\min}(M_k) - \mu_\ell^{\texttt{n1cv2}}) & \text{if } \lambda_{\min}(M_k) > 0. \end{cases}$$

In these relations, $\mu_\ell^{\texttt{qN}}$ is computed using the *reversal quasi-Newton* scalar update in [BGLS03, § 9.3.3]:

$$\mu_\ell^{\texttt{qN}} := \min\left\{ \frac{|\gamma^k - \gamma^{k-1}|^2}{(\gamma^k - \gamma^{k-1})^\top (x^k - x^{k-1})},\ \text{for}\ \begin{array}{l} \gamma^k \in \{Dc_k^\top \hat{G}^{\ell_k}, Dc_k^\top G^{\ell_k}\} \\ \gamma^{k-1} \in \{Dc_k^\top \hat{G}^{\ell_{k-1}}, Dc_k^\top G^{\ell_{k-1}}\} \end{array} \right\},$$

recalling that $\ell_k$ and $\ell_{k-1}$ denote the last two iterations declaring a serious step ($x^k$ and $x^{k-1}$, respectively).

Backtracking steps multiply the current prox-parameter by a factor of two. At consecutive null steps, the prox-parameter is defined as $\mu_{\ell+1} = \max(\mu_\ell, \sqrt{(\mu_\ell + \lambda_{\min}(M_k))\texttt{nul}})$, for $\texttt{nul}$ a counter of null steps. This update satisfies the convergence condition (24). In all cases, if $\lambda_{\min}(M_k) < 0$, $\mu_{\ell+1} \geq -1.1\lambda_{\min}(M_k)$, and $\mu_{\ell+1} \in [\mu_{\min}, \mu_{\max}]$.

### 6.2.2 Parameters for BFGS and Hanso

Both tolerances $\texttt{normtol}$ and $\texttt{evaldist}$ are set to $\sqrt{\texttt{tol}_{\texttt{stop}}}$. In the linesearch, Armijo's and Wolfe's parameters were taken equal to 0.01 and 0.5, respectively. The option for a Strong Wolfe's criterion is not activated, and the method quits when the linesearch fails. As for the quasi-Newton updates, they are of the full memory type, with scaling of the initial Hessian. Finally, the *final bundle* keeps $\min[100, 2n, n+10]$ past gradients.

## 6.3 Benchmark rules

In an effort to make comparisons fair, we adopted the following rules:

– All solvers use the same black-boxes.

– Each solver has a different computational effort per iteration, which depends not only on the solver, but also on how many times the black-box is called per iteration. The number of iterations is not a meaningful measure for comparison, and the number of (bb) calls is different for each solver. While each iteration of HANSO calls ($\mathtt{bb_c}$) and ($\mathtt{bb_h}$), the Composite Bundle calls ($\mathtt{bb_h}$) at Step 2, and can call ($\mathtt{bb_c}$) and ($\mathtt{bb_h}$) at Step 3. Moreover, HANSO always uses gradient values, but the Composite Bundle method requires additional first order information for the inner mapping only at a serious step, to define a new inner model. Striving for fairness, we defined the counters below:

  -For HANSO, one call to both ($\mathtt{bb_c}$) and ($\mathtt{bb_h}$) counted as 1.

  -For the Composite Bundle method, we kept separate counters for the number of $c$, $Dc$, and $h/G$ evaluations: `nc, nDc, nf`, respectively. To each counter corresponds a number of scalars needed for the corresponding calculation: $m$, $mn$, and $1+m$, respectively. This gives the number of scalars required for a single evaluation of $h \circ c$ and a subgradient (as in HANSO):

$$\mathtt{OneEval} = m(1+n) + 1 + m\,.$$

  Accordingly, the number of (bb) calls in the Composite Bundle method was defined as

$$\frac{\mathtt{nc}m + \mathtt{nDc}mn + \mathtt{nf}(1+m)}{\mathtt{OneEval}}\,.$$

  At this point a potential advantage of CBUN becomes clear: when for a given function the method makes many consecutive null steps, this is not expensive in terms of (bb) calls (only ($\mathtt{bb}_h$) is needed).

– We also computed total CPU time for each solver to reach its stopping test. This information should mostly be taken as a complement to the counter of (bb) calls. The reason is that CPU times can be misleading, because for almost all the tested functions the (bb) calls take a negligible time, a feature that is rare in real-life problems. For example, for the nuclear generation planning problems in [ES10, Sec. 7], 95% of the total CPU time is spent in producing the black-box information. By contrast, in all of ours runs and for all but one instance, (bb) calls represent less than 10% of the total CPU times (taking up to 40% for one test-function, written in Fortran and requiring a mex-interface).

– All solvers use the same quadratic programming packages, for which there are two possibilities: a Fortran library with a mex-interface, or a special MATLAB solver. Quadratic programs like (9) can be solved by QUADPROG, the built-in MATLAB QP solver, but we prefer the method in [Kiw86], and made a mex-interface for the Fortran code developed by the author. This method is specially tailored for quadratic minimization over a simplex, as it is the case for the problem dual to (9) and, hence, often outperforms QUADPROG, which is a general solver. HANSO also has a special QP solver, written in MATLAB. Since HANSO QP problems amount to setting $M_k^\ell \equiv 0$ in

$$0 \in conv\{G^i : i \in \mathcal{B}_\ell\} + M_k^\ell d^\ell\,,$$

  which is the optimality condition for (9), we could modify HANSO QP solver to handle (9). Reciprocally, it is possible to use the Fortran QP solver in [Kiw86] to solve HANSO QP problems.

– For large-scale instances, HANSO offers a limited memory BFGS method explained in [Ska10]. However, since for the prox-variable metric the calculation of $M_k$ corresponds to a Newton method, the limited-memory option was not activated in the comparisons.

– Each solver has specific stopping tests, and since BFGS uses a smooth method, the triggers terminating the runs are only heuristical. For each solver we declare a run a success when:

  -For BFGS and HANSO, when the tolerance on the smallest vector in the convex hull of certain subgradients is met (for BFGS, these are the subgradients in its *final bundle*).

  -For CBUN and HYCB, when the stopping test in Step 2 is reached.

– All non-successful runs are declared failures, with a special counter for when a solver reached the maximum number of iterations or calls to the black-box (`maxit` or `maxsim`, denoted by `max` in the tables). Possible reasons for failures are detection of unboundedness in $x$ or in the function values, errors in the QP solver, and, for BFGS, a nondescent direction, or a problem in the linesearch.

– The hybrid variants are not initiated if BFGS detected unboundedness. They are started when BFGS succeeds, reaches `maxit`, cannot descend from the generated direction, or if the linesearch or the QP solver failed.

– Both BFGS and the Composite Bundle use the same starting points, each function was run for 10 different starting points. Unless otherwise specified, the starting points have all components randomly drawn in $[-1, 1]$.
– To measure the accuracy reached by each solver, we only considered test-functions with known optimal value $\bar{f}$, or such that all the solvers converged to the same final value within a tolerance of $10^{-5}$. In this case, the optimal value $\bar{f}$ is given by the smallest value function found by all the solvers. Letting $f^{\text{best}}$ denote the function value of the analyzed case, then

$$RA := -\log_{10} \max\left(10^{-16}, \frac{f^{\text{best}} - \bar{f}}{1 + |\bar{f}|}\right)$$

measures the number of digits of accuracy achieved by the solver.
– We exclude from the tables with results those nonconvex cases for which different solvers found different critical points, see Table 10 in the appendix for details.
– Since full tables are large, for the reader's convenience in this section we only reproduce the overall results of each full table in the Appendix.

## 6.4 Battery of convex test problems

We first consider typical functions for convex NSO benchmarking, described in Table 1.

| Name | $n$ | $\bar{f}$ | Reference |
|---|---|---|---|
| Maxquad | 10 | -0.84140833459641 | [BGLS03, p. 131] |
| BadGuy | 10 | -2048 | [HUL93, p. 277, vol.II Ex.XV.1.1.2] |
| TR48 | 48 | -638565 | [HUL93, p. 21, vol.II, Ex.IX.2.2.6] |
| TSP problems in [HUL93, p. 22, vol.II, Ex.IX.2.2.7], data from TSPLIB95 | | | |
| TSP | 29 | -9015 | `bayg29` |
| TSP | 442 | -50500 | `pcb442` |
| TSP | 1173 | -56349 | `pcb1173` |
| TSP | 3038 | -136601 | `pcb3038` |
| Convex Ury problems, low dimension | | | Ex. 5 with `cubic=0` |
| Ury-cvx | 10 | 500 | |
| Ury-cvx | 20 | 911.833349450300716 | |
| Ury-cvx | 30 | 1118.219919518173128 | |

Table 1: Convex problems in Group 1

The composite structure $h \circ c$ for functions Maxquad, Ury is the one described in the respective examples. Both TR48 and TSP are the piecewise maximum of affine functions, so the inner mapping is affine. However, our Fortran (`bb`) code for TSP was too involved to identify the $c$-components (corresponding to the *minimal 1-trees* in the underlying graph), so we just took the trivial composite case for TSP, as in Remark 1. Similarly for BadGuy, because it does not have a positively homogeneous outer mapping.

Table 2 summarizes the results from Table 8 in the Appendix, obtained for Group 1. In Table 8, each column corresponds to one of the four solvers, CBun, BFGS, Hanso, HyCB, run with the Fortran or Matlab special QP solver (denoted by `f` and `m`, respectively). Since for all of our runs we observed that the Matlab QP special solver is systematically less efficient and/or less reliable than the Fortran one, the shorter Table 2 contains the indicators for the Fortran variants only: CBun`f`, BFGS`f`, Hanso`f`, HyCB`f`.

To each function and each case in Table 1 corresponds a row in Table 8, reporting the figures obtained with each solver, averaged over ten runs, with random starting points. All solvers used the same starting points, with components in $[-1, 1]$ except for BadGuy, taken in $[-512, 512]$. Results are displayed in three columns, with the accuracy RA, the mean CPU time in seconds $\overline{\texttt{sec}}$, and the average number of black-box calls $\overline{(\texttt{bb})}$, respectively. At the end of each function there are two lines. A first line with the mean values mean averaged for all the considered cases (in this line the second column reports between parenthesis the considered number of runs). The second line gives the number of failures and how many of these failures corresponded to having reached the maximum allowed for iterations or simulations (`max`/fails). Finally, the bottom in Table 8 contains the same indicators, averaged over all the problems in the group, as well of the total number of instances considered for the test. These bottom lines and the average of each function are reproduced in Table 2 as a summary.

| `(bb)` Group 1 | | CBunf | | | BFGSf | | | Hansof | | | HyCBf | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | (mean over) | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ |
| BadGuy | (10) | 8 | 1.70 | 135 | 9 | 1.92 | 289 | 9 | 2.53 | 377 | 9 | 2.00 | 301 |
| MQ | (10) | 11 | 0.16 | 40 | 8 | 0.20 | 432 | 8 | 0.32 | 683 | 12 | 0.24 | 443 |
| TR48 | (10) | 12 | 3.51 | 48 | 13 | 32.57 | 2738 | 13 | 290.53 | 29439 | 15 | 33.16 | 2746 |
| TSP | (40) | 8 | 102.05 | 1045 | 13 | 1946.97 | 8188 | 13 | 1986.19 | 18050 | 14 | 2014.92 | 8686 |
| Ury | (30) | 11 | 1.82 | 45 | 9 | 2.64 | 1469 | 11 | 20.03 | 17762 | 15 | 2.72 | 1475 |
| MEAN | (100) | 10 | 22 | 262 | 10 | 397 | 2623 | 11 | 460 | 13262 | 13 | 411 | 2730 |
| max/fails | | | 1/1 | | | 3/38 | | | 0/38 | | | 0/0 | |

Table 2: Group 1 overall results: problems in Table 1.

For Group 1, we observe that all methods are very accurate. When compared to CBun, BFGS exhibits a significant increase both in CPU times and number of `(bb)` calls; failing to trigger its termination criteria 38% of the times (only thrice the reason was `max`). For the TSP family, BFGS is 60% more accurate than CBun, but for getting 5 more digits, BFGS spends 8 (20) times more CPU (`(bb)` calls) than CBun does. We conjecture that endowing TSP with a nontrivial composite structure can improve CBun's average figures (we observed a significant change for TR48, when comparing CBun performance on TR48 black-boxes with and without composite structure). Hanso is the slowest solver, and uses the most `(bb)` calls, but it is not the most accurate method: this hybrid variant did not seem to be adequate for this set of problems, probably because they are all convex. By contrast, HyCB eliminated all of BFGS's failures, with a relatively low additional computational effort: HyCB extra CPU times and `(bb)` calls represent less than 5% of BFGS totals. With respect to CBun, HyCB gain of 30% in accuracy is obtained at the stake of an increase of almost 2000% and 1000% in CPU times and `(bb)` calls. For this group of problems, CBun performs better than all the other solvers.

## 6.5 Convex and nonconvex problems

The two other groups include a mix of convex and nonconvex problems given in Table 3. Group 2 gathers problems with low dimensions ($n \leq 50$), while Group 3 contains high dimensional problems ($n \in \{100, 500\}$). For each instance in Groups 2 and 3, we give the optimal values when known, or the lowest function value found by all solvers, in Table 10 in the Appendix.

For the MQ, EucSum, TiltedNorm, and CPS-collections, matrices and vectors were generated randomly. All $A$-matrices are symmetric positive semidefinite, with condition number equal to $range\, A^2$. The $B$-matrices in CPS are symmetric positive semidefinite and condition number equal to $n^2$. To make calculations possible in our computer, for all the sparse matrices the density was set to $0.1, 0.01, 0.01, 0.001$ for $n = 10, 50, 100, 500$, respectively.

Tables 4 and 5 summarize the results for Group 2 and 3, respectively. In the Appendix, Tables 11 and 12 report the respective full details.

For Group 2 the instances excluded because different solvers found different critical points correspond to two variants of the functions NK and LV, namely F8 and T3 in [Ska10]; and GenModRos with starting point in $[0, 2]$ and $[-2, 2]$.

For this group, the overall results show again that CBun performs better in average that the other three solvers. However, BFGS did better for some instances of LV, as well as for GenModRos and ModRos. CBun had difficulties solving the second instance of LV, corresponding to T3 in [Ska10] with $n = 10$. For the excluded instances of GenModRos, the optimal value (5.337) was found often by BFGS while CBun found only a critical point (with value 9.3283). For the function NesChebRos, nine out of the thirty starting points very difficult to handle by BFGS, but not by CBun, explaining the huge

| Name | Parameters | Reference |
|---|---|---|
| CPS<br>CVX | $n \in \{10, 50, 100, 500\}$<br>$range\, A \in \{0.2, 0.8\}n$ | [Ska10, Sec.4.2.2]<br>$f(x) = \sqrt{x^\top Ax} + x^\top Bx$ |
| MQ<br>CVX | $n \in \{10, 50, 100, 500\}$<br>$range\, A \in \{0.2, 0.8\}n$<br>$m = range\, A + 3$ | Ex. 1,with<br>$c_j(x) = \frac{1}{2}x^\top A_j x + b_j^\top x$<br>$\{A_j\} \geq 0$ and $\{b_j\}_{j=1}^{range\, A}$ LI |
| EucSum<br>NCV | same than MQ, but $n \in \{4, 10, 50, 100\}$ | Ex. 4 with the $\ell_1$-norm, $m_j = 1$, $J = m$, and<br>$\phi_j = c_j$ from MQ |
| TiltedNorm<br>CVX | $n \in \{10, 50, 100, 500\}$<br>$w = 4$ | [Ska10, Sec.4.2.1]<br>$f(x) = w\lvert Ax\rvert + (w-1)e_1^\top Ax$ |
| GenModRos<br>NCV | $n = 12$<br>$U = 1, V = 10$<br>$x^0 \in \Big\{[-0.1, -0.1], [-1, 1], [-2, 0], [0, 2], [-2, 2]\Big\}$ | [Ska10, Sec.4.2.4]<br>$f(x) = \sum_{i=1}^{n-1} \Big(V\frac{i}{n}\lvert x_{i+1} - x_i^2/n\rvert + U\frac{i}{n}(1-x_i)^2\Big)$ |
| ModRos<br>NCV | $n = 2$<br>$w \in \{1, 2, 4, 8\}$ | [LO08, Sec. 5.7]<br>$f(x) = w\lvert x_2 - x_1^2\rvert + (1 - x_2)^2$ |
| NesChebRos<br>NCV | $n \in \{5, 10, 50, 100\}$<br>$x^0 \in [0, 2]$ | [LO08, The nonsmooth variation in Sec. 5.8]<br>$f(x) = \sum_{i=1}^{n-1} \lvert x_{i+1} - 2x_i^2 + 1\rvert + 0.25(x_1 - 1)^2$ |
| Ferrier<br>NCV | $n \in \{10, 50, 100\}$<br>$case \in \{1, 3\}$ | [HS10]<br>$f(x) = \begin{cases} \sum_{i=1}^n \lvert ix_i^2 - 2x_i + \sum_{j=1}^n x_j\rvert & \text{if } case = 1 \\ \max_{i=1}^n \lvert ix_i^2 - 2x_i + \sum_{j=1}^n x_j\rvert & \text{if } case = 3 \end{cases}$ |
| NK<br>CVX/NCV | $n \in \{10, 50, 100, 500\}$<br>$case \in \{1, 3, 4, 5, 8, 9\}$ | Problems F$case$ in [Ska10, Sec.5.4.2]<br>see also [HMM04, Sec. 3] |
| LV<br>NCV | $n \in \{10, 50, 100, 500\}$<br>$case \in \{3, 4, 5, 6\}$ | Problems T$case$ in [Ska10, Sec.5.4.3]<br>see also [LV00] |
| Ury<br>CVX/NCV | $n =\in \{10, 20, 30, 100\}$<br>`cubic` $=\in \{0, 0.01\}$ if $n = 100$, `cubic`$=0.01$ otherwise | Ex. 5 |

Table 3: Convex and nonconvex problems in Groups 2 and 3

difference in accuracy obtained by such solvers. For these problems, we observed that BFGS got stuck at a nonoptimal kink and exited having triggered the heuristical stopping test (the projection of zero on its final bundle was smaller than the tolerance). By contrast, the Rosenbrock modifications GenModRos and ModRos put CBun into trouble: these are the only problems for which CBun systematically makes more `(bb)` evaluations than BFGS. For these functions, CBun finds a very precise minimizer, after taking many serious steps (very short ones); since for each new serious step the mapping Jacobian $Dc(\hat{x}^k)$ is computed, this significantly increases the total `(bb)` counter. Finally, and as observed for Group 1, HyCB seems to be a better hybrid variant than Hanso.

| `(bb)` | Group 2 | CBunf | | | BFGSf | | | Hansof | | | HyCBf | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | (mean over) | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ |
| CPS | (60) | 10 | 0.06 | 10 | 6 | 0.22 | 348 | 6 | 0.31 | 470 | 8 | 0.24 | 352 |
| EucSum | (50) | 9 | 0.72 | 17 | 6 | 0.80 | 574 | 6 | 2.17 | 3708 | 14 | 0.85 | 578 |
| Ferrier | (40) | 6 | 0.46 | 21 | 4 | 0.55 | 777 | 4 | 0.93 | 1775 | 7 | 0.60 | 783 |
| GenModRos | (30) | 10 | 1.52 | 839 | 8 | 0.23 | 472 | 8 | 0.99 | 1821 | 14 | 0.31 | 514 |
| LV | (60) | 8 | 1.42 | 334 | 11 | 1.91 | 1292 | 11 | 8.47 | 6355 | 13 | 1.94 | 1297 |
| MQ | (40) | 9 | 0.14 | 11 | 7 | 3.79 | 2212 | 7 | 20.00 | 9611 | 9 | 3.90 | 2217 |
| ModRos | (40) | 6 | 0.12 | 130 | 6 | 0.03 | 104 | 6 | 0.04 | 124 | 9 | 0.17 | 260 |
| NK | (100) | 11 | 0.51 | 28 | 8 | 0.73 | 577 | 8 | 6.08 | 6195 | 14 | 0.86 | 589 |
| NesChebRos | (30) | 16 | 0.20 | 71 | 1 | 0.70 | 775 | 2 | 8.77 | 10335 | 2 | 1.15 | 1021 |
| TiltedNorm | (30) | 11 | 0.99 | 62 | 7 | 0.12 | 434 | 7 | 0.35 | 1132 | 10 | 0.14 | 439 |
| Ury | (30) | 12 | 1.95 | 58 | 11 | 2.67 | 1444 | 11 | 19.78 | 19460 | 14 | 2.99 | 1456 |
| MEAN | (510) | 10 | 0.73 | 143 | 7 | 1.07 | 819 | 7 | 6.17 | 5544 | 10 | 1.20 | 864 |
| `max`/fails | | | 29/29 | | | 9/76 | | | 0/60 | | | 23/23 | |

Table 4: Group 2 overall results: problems in Table 3, $n \leq 50$.

For Group 3 the instances excluded because different solvers found different critical points correspond to four variants of the functions NK and LV, namely F8 and T3, T5, and T6 in [Ska10]; and EucSum with $n = 100$ and $range\, A_j = 400$.

| (bb) | Group 3 | CBunf | | | BFGSf | | | Hansof | | | HyCBf | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | (mean over) | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ |
| CPS | (60) | 6 | 0.70 | 7 | 5 | 83.31 | 1945 | 5 | 106.62 | 2925 | 7 | 83.64 | 1948 |
| EucSum | (10) | 16 | 0.27 | 8 | 5 | 0.78 | 433 | 5 | 1.08 | 734 | 6 | 0.84 | 435 |
| Ferrier | (20) | 4 | 6.54 | 64 | 10 | 5.83 | 3203 | 10 | 9.41 | 10437 | 10 | 5.89 | 3207 |
| LV | (40) | 4 | 3.97 | 65 | 13 | 101.84 | 8856 | 16 | 232.08 | 29816 | 13 | 102.08 | 8860 |
| MQ | (40) | 4 | 1398.41 | 40 | 3 | 3951.10 | 6646 | 9 | 20483.63 | 33343 | 8 | 5350.40 | 6657 |
| NK | (100) | 9 | 6.49 | 19 | 10 | 85.94 | 5189 | 10 | 195.28 | 21102 | 14 | 87.44 | 5193 |
| NesChebRos | (10) | 16 | 0.84 | 19 | 1 | 29.63 | 5247 | 1 | 112.88 | 44548 | 1 | 29.73 | 5251 |
| TiltedNorm | (10) | 3 | 7.84 | 105 | 7 | 1.59 | 1372 | 7 | 2.04 | 2776 | 7 | 1.62 | 1374 |
| Ury | (20) | 0 | 48.02 | 92 | 12 | 61.73 | 5770 | 12 | 273.85 | 45070 | 16 | 91.74 | 5836 |
| MEAN | (310) | 7 | 163.67 | 47 | 7 | 480.19 | 4295 | 8 | 2379.65 | 21195 | 9 | 639.26 | 4307 |
| max/fails | | | 29/29 | | | 21/53 | | | 0/47 | | | 15/15 | |

Table 5: Group 3 overall results: problems in Table 3, $n = 100$ and $n = 500$.

As expected, this higher dimensional group is more difficult for all the solvers. The low mean $\overline{\texttt{(bb)}}$ for CBun indicates that the methods often stalled making many null/backtracking steps, rather than serious steps. However, fthe second instance of Ferrier functions (corresponding to outer function $h(\cdot) = \max(\cdot)$ and $n = 100$) was difficult for CBun, which made many short serious steps, expensive in terms of (bb) calls. Problem MQ with $n = 500$ and $range\, A_j = 400$ was very difficult to solve by all methods. The 100 runs of the NK family did not seem difficult for any solver. CBun exited problem TiltedNorm having reached the maximum number of iterations, while BFGS found a good point and triggered its heuristical stopping test. For Ury, both CBun and BFGS ended by having reached the maximum number of iterations, but BFGS final point is much better than the one found by CBun. For NesChebRos, BFGS fails in the linesearch, stuck at a nonoptimal kink, and none of the hybrid variants succeeds in getting out of the ridge. For this group, and especially for the nonconvex functions, we see a more erratic behaviour of CBun, even though it still has the best indicators in mean.

## 6.6 Performance Profiles

Figures 2(a), 2(b) and 2(c) contain performance profiles over all the 920 runs, *excluding* cases converging to different critical points, but *including* failures, like in the tables of results. This choice was done not to handicap BFGS, whose heuristical stopping test may fail to be triggered sometimes.

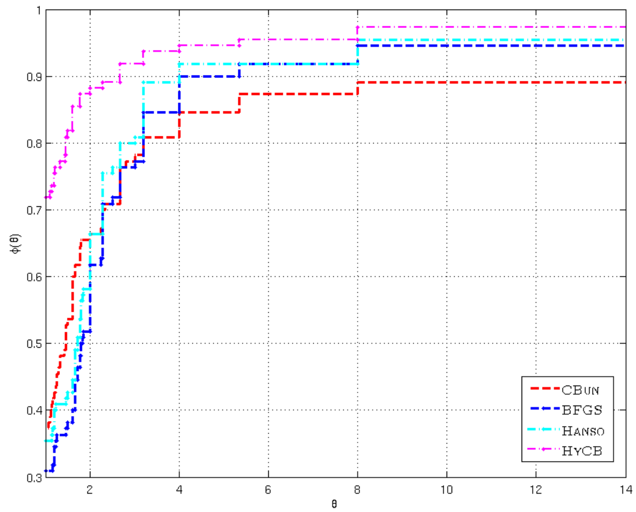
Each line in a performance profile can be interpreted as a cumulative probability distribution of a resource of interest: accuracy, CPU time, (bb) calls. Usually, "smaller" values mean "better performance" of the consider resource. Therefore, for both accuracy and CPU time we plotted the reciprocal of the figures obtained by each solver. In this manner in all the profiles below, the solver with the highest line is the best one for the given indicator of performance. In each profile, we look at the following points:

– The leftmost abscissa values, indicating for each solver the percentage of runs for which each solver had the best performance for the given indicator. The highest value corresponds to the best solver.

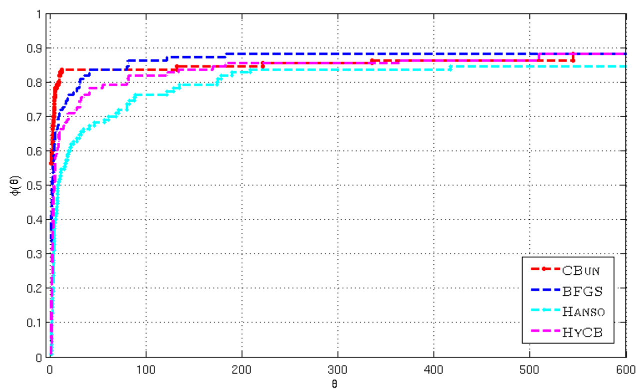
– The abscissa of the intersection between two lines gives the factor that makes the respective solvers comparable.

– For an abscissa value $\theta$ with ordinate $\phi(\theta)$, the value $1 - \phi(\theta)$ corresponds to the fraction of problems that a solver cannot solve within a factor $\theta$ of the best one.

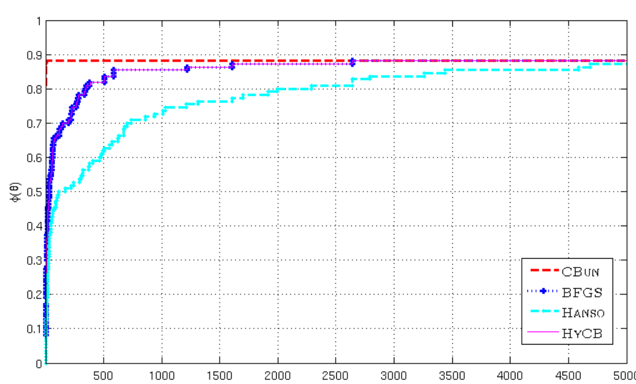
The first profile, in Figure 2(a), shows the performance in terms of accuracy. Looking at the highest value for the leftmost abscissa, we conclude that the hybrid variant HyCB is the most precise solver in 72% of the runs. CBun, Hanso, and BFGS are the most accurate solvers in 37%, 36%, 31% runs, respectively. The abscissa values of 2 show that CBun, BFGS and Hanso failed in 34%, 48% and 34% of the runs to achieve half of HyCB's precision (the respective ordinate values are $\phi(2) = 0.66, 0.52, 0.66$).




(a) (reciprocal of) accuracy.




(b) (reciprocal of) CPU time.




(c) `(bb)` calls.

Figure 2: Performance Profiles.

Profile 2(b) measures the performance in terms of CPU time in seconds, and shows that CBun is the fastest solver in 56% of the runs, followed by BFGS, which was fastest in 33% of the runs. At $\theta = 40$, the solvers ordinates are $\phi(40) = 84, 84, 77, 66$. This means that BFGS is as fast as CBun, within a factor of 40, for 84% of the runs, and takes more than 40 times CBun's CPU time in 16% of the runs. Hanso and HyCB fail to take less than 40 times CBun's CPU time in 23% and 34% of the runs, respectively.

The final profile, in Figure 2(c), measures the performance of the different solvers in terms of `(bb)` calls, and shows a clear superiority of CBun, which appears as the most economic solver in 80% of the cases. Hanso makes extensive use of `(bb)` calls, so it should mostly be used for unstructured nonconvex functions that are not too difficult to evaluate (possibly like the matrix problems in [Ska10, Sec. 5.3]). The lines of BFGS and HyCB practically coincide, making both methods indistinguishable in terms of `(bb)` calls. Since BFGS is faster and HyCB is more precise, the choice between these two solvers should be driven the user's preference (speed or accuracy), keeping in mind that HyCB is more reliable in terms of stopping test.

## 6.7 Determining $\mathcal{V}$-dimension

Many composite functions are partly smooth [Lew02], a notion that generalizes to the nonconvex setting the $\mathcal{VU}$-space decomposition for convex functions in [LOS00] and [MS00].

Identification of the $\mathcal{VU}$ subspaces can be used to determine directions along which the function behaves smoothly and, hence, identify a region where a Newton-like method is likely to succeed. Such smooth directions lie in the $\mathcal{U}$-subspace; its orthogonal complement, the $\mathcal{V}$-subspace, concentrates all the relevant nonsmoothness of the function, at least locally. Near a critical point $\bar{x}$, the $\mathcal{V}$-subpace is spanned by the subdifferential $Dc(\bar{x})^\top \partial h(\bar{C})$, with $\bar{C} = c(\bar{x})$, and the $\mathcal{U}$-subspace is the orthogonal complement of $\mathcal{V}$. Alternatively, in the wording of [Lew02], the $\mathcal{U}$-subspace is the subspace tanget to the *smooth or activity manifold* at $\bar{x}$, and $\mathcal{V} = \mathcal{U}^\perp$.

In [LO08] and [Ska10] it is observed that BFGS can retrieve $\mathcal{VU}$-information by analyzing the eigenvalues of the inverse Hessian used to define a new iterate. For comparison purposes, we consider CBun and BFGS only and estimate the dimension of the respective generated $\mathcal{V}$-subspaces as follows:

– For CBun we compute the range of the subspace spanned by the final strongly active gradients in the bundle:
$$\mathit{dim}\,\mathcal{V}_{\text{CBun}} := \mathit{range}\left\{Dc(x^k)^\top(G^i - \hat{G}^\ell) : i \in \mathcal{B}_\ell \text{ with } \alpha_i^\ell > 0\right\},$$
for $x^k$ the last generated serious step and $\ell$ the iteration triggering the stopping test; recalling that in Step 4 of Algorithm 1 the bundle sizes are kept controlled by a parameter $|\mathcal{B}\max|$,

– For BFGS we count how many eigenvalues of the final inverse Hessian $H$ cluster near 0:
$$\mathit{dim}\,\mathcal{V}_{\text{BFGS}} := \mathit{card}\left\{i \le n : \frac{\lambda_i(H)}{\lambda_{\max}(H)} \le \epsilon\right\},$$
for $\epsilon$ a given tolerance.

| | BadGuy | EucSum | | | | Maxquad | MQ | | | | TR48 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 48 |
| $\mathit{dim}\,\mathcal{V}$ | 10 | 8 | 6 | 4 | 2 | 3 | 8 | 6 | 4 | 2 | ?? |
| CBun | 10 | 10 | 8 | 6 | 4 | 4 | 8 | 7 | 4 | 3 | 47 |
| BFGS | 2.3 | 8 | 3.8 | 3.4 | 2.6 | 3.2 | 8 | 6.1 | 5 | 7 | 47 |

Table 6: $\mathcal{V}$ dimensions for BadGuy, EucSum, Maxquad, MQ, and TR48.

Table 6 reports the obtained results for some of the problems in Group 1 and 2, with low dimension and $\mathcal{V}$-dimensionality depending on the case. The parameters setting were $|\mathcal{B}\max| = 50$ and $\epsilon \in \{0.1, 0.01\}$ (which gave identical results for this group of runs). Each problem was run 10 times with random starting points.

Even though the rules adopted for determining the $\mathcal{V}$-dimensions are rather rough, both CBun and BFGS estimations are reasonable, with a few exceptions. For both solvers the worst results are those obtained for the nonconvex EucSum functions. The extremely low $\mathcal{V}$-dimension estimated by BFGS for BadGuy comes from the fact that it is hard to automatically separate almost null from nonnull, yet small, eigenvalues. An aposteriori (visual) examination of the eigenvalues obtained for each starting point shows a rather erratic behaviour of BFGS for this function over the different starting points, even though BFGS's heuristic stopping test was always triggered. Such oscillation could be explained by a lack of stability of the Hessian with respect to small perturbations, a common phenomenon for a nonsmooth function near a kink.

We made a second group of runs, to determine the impact of smaller or larger $\mathcal{V}$-dimension, with respect to the dimension of the full space. We considered the CPS function, with dimension $n \in \{10, 50, 100\}$ and varying $\mathcal{V}$-dimension. For this example, the $\mathcal{V}$-dimension coincides with the range of the matrix $A$ (taken with sparse density equal to 0.1 for all cases).

Table 7 reports the $\mathcal{V}$-dimensions estimated by CBun and BFGS for different parameters. For CBun, the maximum bundle size was set to 50 and 100: we expect results to be worse if $|\mathcal{B}\max| < dim\,\mathcal{V}$ and the bundle needs to be compressed to an insufficient number of elements. For BFGS, we took two values of $\epsilon$, as before. In the table, the parameters appears between parentheses after the name of each solver.

| | CPS | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | 10 | 10 | 10 | 10 | 50 | 50 | 50 | 50 | 50 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| $dim\,\mathcal{V}$ | 8 | 6 | 4 | 2 | 40 | 30 | 20 | 10 | 2 | 90 | 80 | 70 | 60 | 50 | 40 | 30 | 20 | 10 | 2 |
| CBun(50) | 8 | 6 | 4 | 2 | 40 | 28 | 20 | 10 | 2 | 8 | 46 | 36 | 44 | 44 | 30 | 29 | 20 | 10 | 2 |
| CBun (100) | 8 | 6 | 4 | 2 | 40 | 30 | 20 | 10 | 2 | 77 | 78 | 70 | 56 | 50 | 40 | 30 | 20 | 10 | 2 |
| BFGS$10^{-2}$ | 8 | 6.2 | 4.4 | 2.6 | 44 | 39.5 | 35.5 | 35.5 | 36 | 95.5 | 91 | 86.5 | 88 | 86 | 85 | 86 | 85.5 | 86.5 | 86 |
| BFGS$10^{-3}$ | 8 | 6 | 4 | 2 | 43 | 37 | 31 | 24 | 19 | 93 | 86 | 81 | 80 | 75 | 71 | 67 | 61 | 63 | 64 |

Table 7: $\mathcal{V}$ dimensions for CPS.

We observe that problems with larger $\mathcal{V}$-subspace are more difficult for both solvers. In general, CBun(100) seems to give a reasonable estimate, but this is not always true, especially when $n = 100$.

We conclude our analysis with Figure 3, with the real and estimated $\mathcal{V}$-dimensions for the 30 different functions, showing that in general BFGS overestimates the size of the $\mathcal{V}$-space. We emphasize that this set of tests determining $\mathcal{V}$-dimensionality is only preliminary, and rather crude. For this reason, the conclusions above should not be taken as an indication of goodness or badness of a solver. The subject of determining $\mathcal{VU}$ subspaces is still rather unexplored, with a few exceptions in [LO08], [Ska10], and the MQ functions considered in [DSS09].

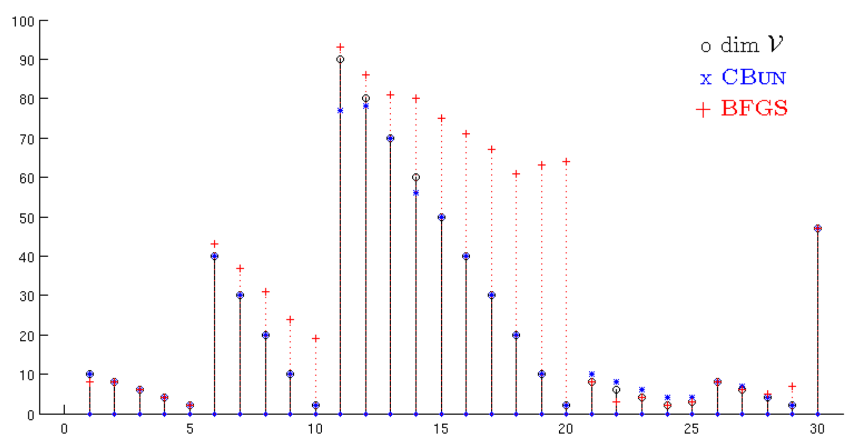



Figure 3: True and estimated $\mathcal{V}$-dimensions for the 30 functions.

## Concluding Remarks

The composite bundle method presented in this work makes tractable the algorithm ProxDescent in [LW08] for a large class of composite functions, with real-valued, positively homogeneous, and convex outer functions. In particular, the method can be applied to minimize some nonconvex nonsmooth functions, a challenging issue for bundle methods. Our composite cutting-planes model, approximating the conceptual model, avoids typical pitfalls in nonconvex bundle methods (recall Figure 1).

The numerical experience reported in this work shows the good performance of the Composite Bundle method for problems of moderate size. For large dimensions, the use of variable prox-metrics may increase the solution times too much, even if there are sparse patterns to exploit. The impact of such increase is problem dependent: for some functions (like CPS and TSP) there is a clear advantage in applying a bundle method ($n \leq 500$ in CPS, and $n \leq 3038$ in TSP, but $M_k \equiv 0$). The advantage is less clear for other functions, especially some of the nonconvex ones in Group 3. Since we sometimes also observed that too many short serious steps made CBun stall, we conjecture that a linesearch (replacing or complementing the curved search modifying $\mu_\ell$) can improve the performance of Algorithm 1 for nonconvex functions, but this is a subject of future research.

Although BFGS is accurate and fast (at least for our examples, with blackboxes computationally light), neither BFGS nor Hanso appeared as the best alternatives for many classes of functions considered in our runs. However, conclusions can be different for a different set of test-functions. Also, the usefulness of a solver depends on the specific purpose sought by the user: since BFGS descends fast from a starting point, it could be an interesting alternative if not much accuracy is required, or if the user seeks for a "better" point, without caring if it is the best one. For some problems, we observed that BFGS got stuck at a nonoptimal kink and exited having triggered the heuristical stopping test (the projection of zero on its final bundle was smaller than the tolerance). If reliability is a concern, the output of BFGS can be plugged into a bundle method, as in HyCB, to satisfy a theoretical stopping test.

However, if too much accuracy is desired, the hybrid variant is likely to increase the computational bulk of BFGS too much (at least when compared to applying directly CBun). As for Hanso, since it makes extensive use of `(bb)` calls, we think it should mostly be used for unstructured nonconvex functions that are not too difficult to evaluate (possibly like the matrix problems in [Ska10, Sec. 5.3]).

Another important issue to consider for a heavy duty application is that, even in the presence of a composite structure, the resulting smooth mapping may be large, or have no special second order sparse patterns to exploit. In this case, it can be sound to use null or diagonal matrices $M_k$ in Algorithm 1, or apply the limited memory variants in [HMM04] and [Ska10].

We mention the work [KBM10], comparing several NSO general purpose solvers for different type and size of problems, as well as for diferent `(bb)` available information. Table 3 therein, analizing the efficiency and reliability of the considered solvers, can be useful as a complement of information to the conclusions drawn in our numerical results, keeping in mind that solvers are different and that the test-functions are not exactly the same, although there is some intersection.

**Comparison with [NPR08].** The proximity control (nonconvex) bundle algorithm [NPR08] considers models for functions such that several Clarke subgradients at one point can be computed at reasonable cost. The proposed scheme is fairly general and bears some resemblance to our composite approach, which we explain next.

Instead of assuming that the objective function enjoys some particular structure, in [NPR08] the authors suppose there is available certain *local model*, $\phi(\cdot, x^k)$, for the objective function $f$ at the current iterate $x^k$. In our notation, $f = h \circ c$, and the local model is $\phi(x^k + \cdot, x^k) = h(c_k(\cdot))$. As explained in [NPR08, Rems. 2.9 and 6.3], such composite model is both a strong and strict first-order model for $f$. The local model is approximated by a *working model*, $\phi_k(\cdot, x)$, which would correspond to our composite cutting-planes model, $\check{h}_\ell(c_k(\cdot))$. However, in [NPR08, Def. 3.3], the first-order working model satisfies the conditions $\phi_k(x, x) = (h \circ c)(x)$ and $\partial_1 \phi_k(x, x) \subset \partial_1 \phi(x, x)$. For our composite model, this would mean to require that

$$\check{h}_\ell(c(x^k)) = (h \circ c)(x^k) \quad \text{and} \quad conv\{G^i \text{ for all } i \in \mathcal{B}_\ell : \check{h}_\ell(c(x^k)) = G^{i\,\top} c(x^k)\} \subset \partial(h \circ c)(x^k)\,,$$

which only holds in our case if the outer subgradient information for $C = c(x^k)$ was kept in the bundle.

Another related important difference is that when building the working model, in addition to (15), (22), and (23), the method in [NPR08] needs to incorporate certain *exact* cutting planes. In our setting, an exact cutting plane would correspond to requiring that

$$\forall \ell \geq 1, \text{ given some } \gamma \in \partial(h \circ c)(x^k), \quad (h \circ c)(x^k) + \gamma^\top \cdot \leq \check{h}_\ell(c_k(\cdot))\,.$$

Instead, in (13) we use the outer subgradient $\Gamma^\ell \in \partial h(C^\ell)$ for $C^\ell = c(x^k + d^\ell)$ to detect if the linearization of the inner mapping is not good enough and trigger the backtracking process. But if $x^k + d^\ell$ is declared a serious step, the corresponding subgradient $\Gamma^\ell$ does not enter the bundle (nothing prevents the bundle management step to incorporate this data, though).

Like ours, the quadratic programming subproblem in [NPR08] includes a second-order term with a possibly nonpositive definite matrix $M_k$, augmented by a (positive enough) matrix $\mu_\ell \mathrm{I}$. But the acceptance test corresponding to (12) uses a predicted decrease that is different from ours. Namely, instead of $\delta_\ell$ in (11), the larger amount $\delta'_\ell := \delta_\ell + \frac{1}{2}\mu_\ell d^{\ell\,\top} d^\ell$ is used. As for null steps, the decision on whether or not to increase the parameter $\mu_\ell$ is done by checking if, for some parameter $m_3 \in (m_1, 1)$,

$$h(c(x^k) + D_k d^\ell) \leq (h \circ c)(x^k) - m_3 \delta'_\ell$$

(the prox-parameter is left unchanged if the inequality above does not hold).

Convergence results for the proximity control bundle method with strong first-order models are similar to ours. The method keeps matrices $M_k$ bounded below and above by $\pm q\mathrm{I}$ for some $0 < q < +\infty$, so (18) always holds. The case of infinite null steps is treated in [NPR08, Lem. 4.1], where it is shown that a subsequence of the prox-parameter diverges and, hence, checking satisfaction of our condition (24) is not straightforward.

Section 7 in [NPR08], devoted to applications, contains several cases showing the good numerical behaviour of the algorithm. An interesting subject of future research would be to compare both algorithms performance on composite objective functions.

# References


[BF95] J. V. Burke and M. C. Ferris, *A Gauss-Newton method for convex composite optimization*, Mathematical Programming **71** (1995), 179–194.

[BGLS03] J.F. Bonnans, J.Ch. Gilbert, C. Lemaréchal, and C. Sagastizábal, *Numerical optimization. theoretical and practical aspects*, Universitext, Springer-Verlag, Berlin, 2003, xiv+423 pp.

[BLO02] J.V. Burke, A. Lewis, and M. Overton, *Two numerical methods for optimizing matrix stability*, Linear Algebra and its Applications **351-352** (2002), 117–145.

[BS00] J. F. Bonnans and A. Shapiro, *Perturbation analysis of optimization problems*, Springer Series in Operations Research, Springer-Verlag, New York, 2000.

[CL93] R. Correa and C. Lemaréchal, *Convergence of some algorithms for convex minimization*, Math. Program. **62** (1993), no. 2, 261–275.

[DSS09] Aris Daniilidis, Claudia Sagastizábal, and Mikhail Solodov, *Identifying structure of nonsmooth convex functions by the bundle technique*, SIAM Journal on Optimization **20** (2009), no. 2, 820–840.

[ES10] G. Emiel and C. Sagastizábal, *Incremental-like bundle methods with application to energy planning*, Computational Optimization and Applications **46** (2010), 305–332.

[Fle87] R. Fletcher, *Practical methods of optimization* (second edition), John Wiley & Sons, Chichester, 1987.

[Har03] W.L. Hare, *Nonsmoth Optimization with Smooth Substructure*, Ph.D. thesis, Department of Mathematics, Simon Fraser University, 2003.

[HMM04] M. Haarala, K. Miettinen, and M. M. Makela, *New limited memory bundle method for large-scale nonsmooth optimization*, Optimization Methods and Software (2004), no. 6, 673–692.

[HS10] Warren Hare and Claudia Sagastizábal, *A redistributed proximal bundle method for nonconvex optimization*, SIAM Journal on Optimization **20** (2010), no. 5, 2442–2473.

[HUL93] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex analysis and minimization algorithms*, Grund. der math. Wiss, no. 305-306, Springer-Verlag, 1993, (two volumes).

[HUL01] J.-B. Hiriart-Urruty and C. Lemaréchal, *Fundamentals of convex analysis*, Grundlehren Text Editions, Springer-Verlag, Berlin, 2001.

[KBM10] N. Karmitsa, A. Bagirov, and M. M. Makela, *Comparing different nonsmooth minimization methods and software*, Optimization Methods and Software (2010).

[Kiw86] K.C. Kiwiel, *A method for solving certain quadratic programming problems arising in nonsmooth optimization*, IMA Journal of Numerical Analysis **6** (1986), 137–152.

[Lew02] A.S. Lewis, *Active sets, nonsmoothness and sensitivity*, SIAM Journal on Optimization **13** (2002), no. 3, 702–725.

[LO08] Adrian S. Lewis and Michael L. Overton, *Nonsmooth Optimization via BFGS*, Avalaible at `http://www.optimization-online.org/DB_HTML/2008/12/2172.html`., 2008.

[LOS00] C. Lemaréchal, F. Oustry, and C. Sagastizábal, *The $\mathcal{U}$-Lagrangian of a convex function*, Trans. Amer. Math. Soc. **352** (2000), no. 2, 711–729.

[LS97] C. Lemaréchal and C. Sagastizábal, *Practical aspects of the Moreau-Yosida regularization: theoretical preliminaries*, SIAM Journal on Optimization **7** (1997), no. 2, 367–385.

[LV00] L. Luksan and J. Vlcek, *Test problems for nonsmooth unconstrained and linearly constrained optimization*, Tech. Report 798, Institute of Computer Science, Academy of Sciences of the Czech Republic, 2000.

[LW02] Chong Li and Xinghua Wang, *On convergence of the gauss-newton method for convex composite optimization*, Mathematical Programming **91** (2002), 349–356, 10.1007/s101070100249.

[LW08] A.S. Lewis and S.J. Wright, *A proximal method for composite minimization*, Available at `http://www.optimization-online.org/DB_HTML/2008/12/2162.html`, 2008.

[Mif77] R. Mifflin, *Semi-smooth and semi-convex functions in constrained optimization*, SIAM Journal on Control and Optimization **15** (1977), 959–972.

[MS00] R. Mifflin and C. Sagastizábal, *On $\mathcal{VU}$-theory for functions with primal-dual gradient structure*, SIAM Journal on Optimization **11** (2000), no. 2, 547–571.

[MS03] ______, *Primal-Dual Gradient Structured Functions: second-order results; links to epiderivatives and partly smooth functions*, SIAM Journal on Optimization **13** (2003), no. 4, 1174–1194.

[MS04] ______, *$\mathcal{VU}$-Smoothness and Proximal Point Results for Some Nonconvex Functions*, Optimization Methods and Software **19** (2004), no. 5, 463–478.

[MS05] ______, *A $\mathcal{V}U$-algorithm for convex minimization*, Math. Program., Ser. A **104** (2005), no. 2–3, 583–608.

[Nes07] Y. Nesterov, *Gradient methods for minimizing composite objective functions*, Discussion paper 2007/76, CORE, 2007, Avalaible at `http://www.optimization-online.org/DB_HTML/2007/09/1784.html`.

[NPR08] D. Noll, O. Prot, and A. Rondepierre, *A proximity control algorithm to minimize nonsmooth and nonconvex functions*, Pac. J. Optim. **4** (2008), no. 3, 569 – 602.

[NY83] A. Nemirovsky and D. Yudin, *Problem complexity and method efficiency in optimization*, A Wiley-Interscience Publication, John Wiley & Sons Inc., New York, 1983.

[Ous00] F. Oustry, *A second-order bundle method to minimize the maximum eigenvalue function*, Math. Program. **89** (2000), no. 1, Ser. A, 1–33.

[RW98] R.T. Rockafellar and R.J.-B. Wets, *Variational analysis*, Grund. der math. Wiss, no. 317, Springer-Verlag, 1998.

[Sha03] A. Shapiro, *On a class of nonsmooth composite functions*, Math. Oper. Res. **28** (2003), no. 4, 677–692.

[Ska10] Anders Skajaa, *Limited Memory BFGS for Nonsmooth Optimization*, Master's thesis, Courant Institute of Mathematical Science, 2010.

Appendix with full tables

| (bb) | #-$n$ | CBUNf | | | CBUNm | | | BFGSf | | | BFGSm | | | HANSOf | | | HANSOm | | | HYCBf | | | HYCBm | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | RA, | $\overline{\text{sec}}$, | $\overline{\text{(bb)}}$ | RA, | $\overline{\text{sec}}$, | $\overline{\text{(bb)}}$ | RA, | $\overline{\text{sec}}$, | $\overline{\text{(bb)}}$ | RA, | $\overline{\text{sec}}$, | $\overline{\text{(bb)}}$ | RA, | $\overline{\text{sec}}$, | $\overline{\text{(bb)}}$ | RA, | $\overline{\text{sec}}$, | $\overline{\text{(bb)}}$ | RA, | $\overline{\text{sec}}$, | $\overline{\text{(bb)}}$ | RA, | $\overline{\text{sec}}$, | $\overline{\text{(bb)}}$ |
| BadGuy | 1-10 | 8, | 1.70, | 135 | 1, | 24.32, | 1501 | 9, | 1.92, | 289 | 9, | 1.94, | 289 | 9, | 2.53, | 377 | 9, | 2.52, | 378 | 9, | 2.00, | 301 | 9, | 22.12, | 1793 |
| mean, | (10) | 8, | 1.70, | 135 | 1, | 24.32, | 1501 | 9, | 1.92, | 289 | 9, | 1.94, | 289 | 9, | 2.53, | 377 | 9, | 2.52, | 378 | 9, | 2.00, | 301 | 9, | 22.12, | 1793 |
| max/fails | | | 0/0 | | | 10/10 | | | 0/0 | | | 0/0 | | | 0/0 | | | 0/0 | | | 0/0 | | | 10/10 | |
| MQ | 1-10 | 11, | 0.16, | 40 | 11, | 1.20, | 62 | 8, | 0.20, | 432 | 8, | 0.77, | 441 | 8, | 0.32, | 683 | 8, | 1.66, | 1451 | 12, | 0.24, | 443 | 12, | 0.84, | 452 |
| mean, | (10) | 11, | 0.16, | 40 | 11, | 1.20, | 62 | 8, | 0.20, | 432 | 8, | 0.77, | 441 | 8, | 0.32, | 683 | 8, | 1.66, | 1451 | 12, | 0.24, | 443 | 12, | 0.84, | 452 |
| max/fails | | | 0/0 | | | 0/0 | | | 0/1 | | | 0/1 | | | 0/0 | | | 0/1 | | | 0/0 | | | 0/0 | |
| TR48 | 1-48 | 12, | 3.51, | 48 | 3, | 133.67, | 196 | 13, | 32.57, | 2738 | 13, | 46.02, | 2738 | 13, | 290.53, | 29439 | 13, | 232.43, | 20537 | 15, | 33.16, | 2746 | 13, | 115.59, | 2816 |
| mean, | (10) | 12, | 3.51, | 48 | 3, | 133.67, | 196 | 13, | 32.57, | 2738 | 13, | 46.02, | 2738 | 13, | 290.53, | 29439 | 13, | 232.43, | 20537 | 15, | 33.16, | 2746 | 13, | 115.59, | 2816 |
| max/fails | | | 0/0 | | | 10/10 | | | 0/10 | | | 0/10 | | | 0/10 | | | 0/10 | | | 0/0 | | | 4/4 | |
| | 1-29 | 16, | 0.11, | 50 | 6, | 9.68, | 608 | 9, | 0.12, | 173 | 9, | 0.09, | 173 | 9, | 0.20, | 319 | 9, | 0.19, | 313 | 9, | 0.12, | 180 | 9, | 0.10, | 180 |
| TSP | 2-442 | 7, | 13.38, | 996 | 7, | 20.30, | 1000 | 16, | 60.39, | 5145 | 16, | 68.11, | 5145 | 16, | 217.17, | 44446 | 16, | 241.89, | 44446 | 16, | 66.89, | 5660 | 16, | 77.13, | 5664 |
| | 3-1173 | 6, | 91.09, | 2044 | 6, | 83.85, | 1363 | 16, | 352.03, | 5657 | 16, | 360.70, | 5657 | 16, | 352.03, | 5657 | 16, | 360.70, | 5657 | 16, | 374.85, | 6202 | 16, | 391.58, | 6216 |
| | 4-3038 | 4, | 303.63, | 1090 | 4, | 330.90, | 1092 | 10, | 7375.32, | 21777 | 10, | 7352.68, | 21777 | 10, | 7375.33, | 21777 | 10, | 7352.68, | 21777 | 16, | 7617.81, | 22701 | 11, | 7586.84, | 22592 |
| mean, | (40) | 8, | 102.05, | 1045 | 6, | 111.18, | 1016 | 13, | 1946.97, | 8188 | 13, | 1945.40, | 8188 | 13, | 1986.19, | 18050 | 13, | 1988.87, | 18048 | 14, | 2014.92, | 8686 | 13, | 2013.91, | 8663 |
| max/fails | | | 1/1 | | | 0/0 | | | 2/6 | | | 2/6 | | | 0/6 | | | 0/6 | | | 0/0 | | | 0/0 | |
| | 1-10 | 11, | 0.18, | 18 | 0, | 66.20, | 384 | 8, | 0.51, | 578 | 14, | 3.40, | 823 | 14, | 7.39, | 9855 | 14, | 6.60, | 3932 | 16, | 0.53, | 584 | 14, | 13.20, | 961 |
| Ury | 3-20 | 12, | 1.08, | 39 | 0, | 175.27, | 415 | 13, | 2.12, | 1395 | 13, | 10.13, | 1395 | 13, | 20.29, | 19696 | 13, | 25.20, | 8770 | 14, | 2.18, | 1401 | 16, | 14.59, | 1430 |
| | 5-30 | 10, | 4.20, | 77 | -1, | 199.46, | 285 | 7, | 5.29, | 2433 | 7, | 41.38, | 2503 | 7, | 32.42, | 23734 | 7, | 82.82, | 18543 | 16, | 5.44, | 2440 | 12, | 84.60, | 2635 |
| mean, | (30) | 11, | 1.82, | 45 | -0, | 146.98, | 362 | 9, | 2.64, | 1469 | 11, | 18.30, | 1574 | 11, | 20.03, | 17762 | 11, | 38.20, | 10415 | 15, | 2.72, | 1475 | 14, | 37.46, | 1675 |
| max/fails | | | 0/0 | | | 30/30 | | | 1/21 | | | 1/29 | | | 0/22 | | | 0/29 | | | 0/0 | | | 6/6 | |
| MEAN, | (100) | 10, | 22, | 262 | 4, | 83, | 627 | 10, | 397, | 2623 | 11, | 402, | 2646 | 11, | 460, | 13262 | 11, | 453, | 10166 | 13, | 411, | 2730 | 12, | 438, | 3080 |
| max/fails | | | 1/1 | | | 50/50 | | | 3/38 | | | 3/46 | | | 0/38 | | | 0/46 | | | 0/0 | | | 20/20 | |

Table 8: Results for Group 1: problems in Table 1.

| Name | Parameters | $\overline{h \circ c}$ |
|---|---|---|
| CPS | $n \in \{10, 50, 100, 500\}$ | 0 (opt) |
| CVX | $range\, A \in \{0.2, 0.8\}n$ | |
| MQ | $n \in \{10, 50, 100, 500\}$ | 0 (opt) |
| CVX | $range\, A \in \{0.2, 0.8\}n$ | |
| | $m = range\, A + 3$ | |
| EucSum | | |
| NCV | $n = 4, range\, A = 2$ | 0.930538450443740 (best) |
| NCV | $n = 10, range\, A = 8$ | 0.465587005455171 (best) |
| NCV | $n = 10, range\, A = 2$ | 0.666424291184390 (best) |
| NCV | $n = 50, range\, A = 40$ | 0.399571129728750 (best) |
| NCV | $n = 50, range\, A = 10$ | 0.002143493387185 (best) |
| NCV | $n = 100, range\, A = 80$ | excluded |
| NCV | $n = 100, range\, A = 20$ | 0.333869560359649 (best) |
| TiltedNorm | $n \in \{10, 50, 100, 500\}$ | 0 (opt) |
| CVX | $w = 4$ | |
| GenModRos | $n = 12$ | 5.377690121369670 (best) |
| NCV | $U = 1, V = 10$ | |
| ModRos | $n = 2$ | 0 (opt) |
| NCV | $w \in \{1, 2, 4, 8\}$ | |
| NesChebRos | $n \in \{5, 10, 50, 100\}$ | 0 (opt) |
| NCV | $x^0 \in [0, 2]$ | |
| Ferrier | $n \in \{10, 50, 100\}$ | 0 (opt) |
| NCV | $case \in \{1, 3\}$ | |
| NK | | |
| NCV | $case = 8$ | excluded |
| CVX/NCV | $case \in \{1, 3, 4, 5, 9\}$ | $\{0, -\sqrt{2}(n-1), 2(n-1), 2(n-1), 0\}$ (opt) |
| LV | | |
| NCV | $case = 3,\ n \in \{10, 50, 100, 500\}$ | excluded |
| NCV | $case = 4,\ n = 10$ | 106.059118520625645 (best) |
| NCV | $case = 4,\ n = 50$ | 587.997761620671213 (best) |
| NCV | $case = 4,\ n = 100$ | 1190.421065495747825 (best) |
| NCV | $case = 4,\ n = 500$ | 6009.807496496680869 (best) |
| NCV | $case \in \{5, 6\},\ n \leq 100$ | 0 (best) |
| NCV | $case \in \{5, 6\},\ n = 500$ | (excluded) |
| Ury | | |
| NCV | $n = 10$, `cubic`=0.01 | 500 (best) |
| NCV | $n = 20$, `cubic`=0.01 | 909.889558838787480 (best) |
| NCV | $n = 30$, `cubic`=0.01 | 1114.734712066170232 (best) |
| CVX | $n = 100$, `cubic`=0 | 1159.869805021747879 (best) |
| NCV | $n = 100$, `cubic`=0.01 | 1162.455887489049701 (best) |

Table 10: Optimal (opt) or best (best) function values, for problems in Groups 2 and 3.

| (bb) | #-n | CBunf | | | BFGSf | | | HANSOf | | | HyCBf | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ |
| CPS | 1-10 | 12 | 0.07 | 14 | 7 | 0.07 | 184 | 7 | 0.11 | 245 | 8 | 0.09 | 187 |
| | 2-10 | 12 | 0.03 | 14 | 7 | 0.05 | 131 | 7 | 0.09 | 196 | 9 | 0.06 | 134 |
| | 3-10 | 11 | 0.02 | 8 | 6 | 0.04 | 94 | 6 | 0.07 | 156 | 9 | 0.05 | 98 |
| | 4-50 | 6 | 0.07 | 9 | 5 | 0.51 | 772 | 5 | 0.65 | 953 | 5 | 0.54 | 774 |
| | 5-50 | 6 | 0.08 | 9 | 6 | 0.39 | 560 | 6 | 0.53 | 741 | 10 | 0.42 | 563 |
| | 6-50 | 10 | 0.07 | 8 | 6 | 0.27 | 349 | 6 | 0.41 | 530 | 9 | 0.29 | 352 |
| mean, | (60) | 10 | 0.06 | 10 | 6 | 0.22 | 348 | 6 | 0.31 | 470 | 8 | 0.24 | 352 |
| max/fails | | | 0/0 | | | 0/0 | | | 0/0 | | | 0/0 | |
| EucSum | 1-4 | 10 | 0.34 | 9 | 6 | 0.08 | 40 | 6 | 0.12 | 65 | 16 | 0.10 | 43 |
| | 2-10 | 7 | 0.09 | 27 | 7 | 0.21 | 590 | 7 | 6.36 | 15622 | 16 | 0.25 | 601 |
| | 3-10 | 16 | 0.01 | 6 | 5 | 0.02 | 75 | 5 | 0.06 | 143 | 5 | 0.03 | 77 |
| | 4-50 | 5 | 3.02 | 30 | 6 | 3.35 | 1411 | 6 | 3.79 | 1626 | 16 | 3.49 | 1416 |
| | 5-50 | 6 | 0.13 | 11 | 7 | 0.34 | 752 | 7 | 0.50 | 1085 | 16 | 0.36 | 754 |
| mean, | (50) | 9 | 0.72 | 17 | 6 | 0.80 | 574 | 6 | 2.17 | 3708 | 14 | 0.85 | 578 |
| max/fails | | | 1/1 | | | 0/0 | | | 0/0 | | | 0/0 | |
| Ferrier | 1-10 | 8 | 0.04 | 8 | 4 | 0.11 | 142 | 4 | 0.16 | 254 | 7 | 0.12 | 148 |
| | 2-10 | 7 | 0.02 | 9 | 4 | 0.03 | 141 | 5 | 0.11 | 348 | 9 | 0.05 | 147 |
| | 3-50 | 5 | 0.26 | 13 | 2 | 1.65 | 1865 | 2 | 2.76 | 4820 | 6 | 1.76 | 1874 |
| | 4-50 | 5 | 1.51 | 53 | 4 | 0.39 | 961 | 5 | 0.69 | 1678 | 5 | 0.46 | 965 |
| mean, | (40) | 6 | 0.46 | 21 | 4 | 0.55 | 777 | 4 | 0.93 | 1775 | 7 | 0.60 | 783 |
| max/fails | | | 0/0 | | | 0/5 | | | 0/0 | | | 0/0 | |
| GenModRos | 1-12 | 9 | 1.93 | 1050 | 7 | 0.24 | 489 | 7 | 1.10 | 2044 | 16 | 0.35 | 547 |
| | 2-12 | 6 | 2.19 | 1229 | 8 | 0.21 | 447 | 8 | 0.83 | 1568 | 16 | 0.32 | 503 |
| | 3-12 | 16 | 0.43 | 237 | 9 | 0.24 | 480 | 9 | 1.03 | 1850 | 10 | 0.26 | 492 |
| | 4-12 | $\neq$ stationary points | | | | | | | | | | | |
| | 5-12 | $\neq$ stationary points | | | | | | | | | | | |
| mean, | (30) | 10 | 1.52 | 839 | 8 | 0.23 | 472 | 8 | 0.99 | 1821 | 14 | 0.31 | 514 |
| max/fails | | | 2/2 | | | 0/0 | | | 0/0 | | | 0/0 | |
| LV | 1-10 | $\neq$ stationary points | | | | | | | | | | | |
| | 2-10 | 4 | 4.70 | 1740 | 8 | 0.17 | 347 | 8 | 1.52 | 2639 | 16 | 0.18 | 351 |
| | 3-10 | 16 | 0.02 | 8 | 5 | 0.09 | 206 | 5 | 0.29 | 479 | 6 | 0.11 | 209 |
| | 4-10 | 16 | 0.03 | 15 | 4 | 0.15 | 305 | 5 | 0.24 | 453 | 7 | 0.16 | 311 |
| | 5-50 | $\neq$ stationary points | | | | | | | | | | | |
| | 6-50 | 3 | 1.89 | 149 | 16 | 5.16 | 2534 | 16 | 42.32 | 29835 | 16 | 5.22 | 2539 |
| | 7-50 | 3 | 1.21 | 55 | 16 | 2.07 | 1582 | 16 | 2.33 | 1764 | 16 | 2.11 | 1586 |
| | 8-50 | 5 | 0.65 | 34 | 16 | 3.84 | 2778 | 16 | 4.10 | 2960 | 16 | 3.88 | 2783 |
| mean, | (60) | 8 | 1.42 | 334 | 11 | 1.91 | 1292 | 11 | 8.47 | 6355 | 13 | 1.94 | 1297 |
| max/fails | | | 10/10 | | | 0/10 | | | 0/10 | | | 0/0 | |
| MQ | 2-10 | 10 | 0.06 | 13 | 8 | 0.30 | 680 | 9 | 1.66 | 3755 | 10 | 0.32 | 687 |
| | 3-10 | 10 | 0.02 | 10 | 7 | 0.04 | 144 | 7 | 0.07 | 205 | 10 | 0.05 | 147 |
| | 4-50 | 7 | 0.44 | 11 | 4 | 13.74 | 5769 | 4 | 76.94 | 31841 | 8 | 14.11 | 5775 |
| | 5-50 | 9 | 0.06 | 8 | 8 | 1.08 | 2256 | 8 | 1.31 | 2642 | 9 | 1.13 | 2259 |
| mean, | (40) | 9 | 0.14 | 11 | 7 | 3.79 | 2212 | 7 | 20.00 | 9611 | 9 | 3.90 | 2217 |
| max/fails | | | 0/0 | | | 7/20 | | | 0/10 | | | 0/0 | |
| ModRos | 1-2 | 9 | 0.07 | 38 | 6 | 0.02 | 37 | 6 | 0.03 | 68 | 10 | 0.04 | 67 |
| | 2-2 | 7 | 0.04 | 53 | 6 | 0.02 | 56 | 6 | 0.03 | 80 | 10 | 0.06 | 100 |
| | 3-2 | 4 | 0.13 | 150 | 6 | 0.03 | 109 | 6 | 0.03 | 122 | 9 | 0.27 | 382 |
| | 4-2 | 2 | 0.24 | 278 | 5 | 0.06 | 214 | 5 | 0.07 | 228 | 6 | 0.30 | 492 |
| mean, | (40) | 6 | 0.12 | 130 | 6 | 0.03 | 104 | 6 | 0.04 | 124 | 9 | 0.17 | 260 |
| max/fails | | | 13/13 | | | 0/0 | | | 0/0 | | | 17/17 | |
| NK | 1-10 | 9 | 0.10 | 32 | 6 | 0.03 | 63 | 6 | 0.07 | 126 | 16 | 0.08 | 81 |
| | 10-50 | 16 | 0.55 | 23 | 8 | 0.06 | 75 | 8 | 0.20 | 298 | 11 | 0.17 | 79 |
| | 11-50 | $\neq$ stationary points | | | | | | | | | | | |
| | 12-50 | 16 | 0.11 | 8 | 5 | 0.04 | 72 | 5 | 0.15 | 269 | 10 | 0.13 | 76 |
| | 2-10 | 16 | 0.30 | 82 | 6 | 0.07 | 191 | 7 | 0.17 | 408 | 9 | 0.32 | 237 |
| | 3-10 | 13 | 0.05 | 12 | 6 | 0.07 | 199 | 6 | 0.34 | 741 | 16 | 0.09 | 205 |
| | 4-10 | 14 | 0.09 | 31 | 7 | 0.02 | 70 | 7 | 0.06 | 140 | 16 | 0.05 | 77 |
| | 5-10 | $\neq$ stationary points | | | | | | | | | | | |
| | 6-10 | 11 | 0.03 | 12 | 5 | 0.02 | 76 | 5 | 0.06 | 141 | 16 | 0.06 | 88 |
| | 7-50 | 8 | 0.44 | 22 | 5 | 0.09 | 238 | 5 | 0.18 | 431 | 16 | 0.43 | 251 |
| | 8-50 | 5 | 3.01 | 44 | 16 | 3.14 | 2351 | 16 | 23.55 | 29652 | 16 | 3.32 | 2357 |
| | 9-50 | 6 | 0.45 | 11 | 16 | 3.79 | 2438 | 16 | 36.04 | 29739 | 16 | 3.98 | 2443 |
| mean, | (100) | 11 | 0.51 | 28 | 8 | 0.73 | 577 | 8 | 6.08 | 6195 | 14 | 0.86 | 589 |
| max/fails | | | 0/0 | | | 0/19 | | | 0/19 | | | 1/1 | |
| NesChebRos | 1-5 | 16 | 0.36 | 188 | 1 | 0.16 | 415 | 2 | 0.58 | 1886 | 2 | 0.45 | 614 |
| | 2-10 | 16 | 0.02 | 9 | 2 | 0.08 | 184 | 2 | 0.15 | 398 | 2 | 1.09 | 716 |
| | 3-50 | 16 | 0.21 | 14 | 1 | 1.87 | 1727 | 1 | 25.59 | 28720 | 1 | 1.91 | 1732 |
| mean, | (30) | 16 | 0.20 | 71 | 1 | 0.70 | 775 | 2 | 8.77 | 10335 | 2 | 1.15 | 1021 |
| max/fails | | | 3/3 | | | 2/3 | | | 0/2 | | | 5/5 | |
| TiltedMax | 1-10 | 14 | 0.10 | 30 | 6 | 0.05 | 180 | 6 | 0.13 | 407 | 15 | 0.07 | 186 |
| | 2-20 | 13 | 0.25 | 46 | 7 | 0.09 | 358 | 7 | 0.30 | 1043 | 9 | 0.12 | 365 |
| | 3-50 | 6 | 2.63 | 109 | 7 | 0.22 | 763 | 7 | 0.60 | 1946 | 7 | 0.24 | 766 |

Continued on next page

| (bb) | #-$n$ | CBUNf | | | BFGSf | | | HANSOf | | | HyCBf | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ |
| mean, | (30) | 11 | 0.99 | 62 | 7 | 0.12 | 434 | 7 | 0.35 | 1132 | 10 | 0.14 | 439 |
| max/fails | | | 0/0 | | | 0/0 | | | 0/0 | | | 0/0 | |
| | 2-10 | 16 | 0.18 | 20 | 7 | 0.29 | 399 | 7 | 10.16 | 14847 | 10 | 0.30 | 405 |
| Ury | 4-20 | 11 | 1.01 | 42 | 11 | 1.90 | 1305 | 11 | 18.19 | 19606 | 16 | 2.06 | 1316 |
| | 6-30 | 10 | 4.66 | 110 | 16 | 5.81 | 2627 | 16 | 30.98 | 23928 | 16 | 6.61 | 2649 |
| mean, | (30) | 12 | 1.95 | 58 | 11 | 2.67 | 1444 | 11 | 19.78 | 19460 | 14 | 2.99 | 1456 |
| max/fails | | | 0/0 | | | 0/19 | | | 0/19 | | | 0/0 | |
| MEAN, | (510) | 10 | 0.73 | 143 | 7 | 1.07 | 819 | 7 | 6.17 | 5544 | 10 | 1.20 | 864 |
| max/fails | | | 29/29 | | | 9/76 | | | 0/60 | | | 23/23 | |

Table 11: Results for Group 2: problems in Table 3, $n \leq 50$.

| (bb) | #-$n$ | CBUNf | | | BFGSf | | | HANSOf | | | HyCBf | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ | RA | $\overline{\texttt{sec}}$ | $\overline{\texttt{(bb)}}$ |
| | 10-500 | 6 | 1.31 | 7 | 3 | 106.27 | 2409 | 3 | 227.11 | 6780 | 3 | 106.60 | 2410 |
| | 11-500 | 5 | 1.26 | 7 | 5 | 229.58 | 4214 | 5 | 238.07 | 4515 | 8 | 230.25 | 4217 |
| CPS | 12-500 | 5 | 1.25 | 7 | 5 | 152.19 | 2011 | 5 | 160.79 | 2312 | 9 | 152.93 | 2014 |
| | 7-100 | 6 | 0.13 | 7 | 5 | 5.27 | 1464 | 5 | 5.95 | 1765 | 5 | 5.33 | 1466 |
| | 8-100 | 6 | 0.14 | 8 | 5 | 4.26 | 1005 | 5 | 4.85 | 1306 | 9 | 4.34 | 1008 |
| | 9-100 | 6 | 0.14 | 8 | 6 | 2.29 | 569 | 6 | 2.95 | 870 | 10 | 2.36 | 572 |
| mean, | (60) | 6 | 0.70 | 7 | 5 | 83.31 | 1945 | 5 | 106.62 | 2925 | 7 | 83.64 | 1948 |
| max/fails | | | 0/0 | | | 0/0 | | | 0/0 | | | 0/0 | |
| EucSum | 6-100 | $\neq$ stationary points | | | | | | | | | | | |
| | 7-100 | 16 | 0.27 | 8 | 5 | 0.78 | 433 | 5 | 1.08 | 734 | 6 | 0.84 | 435 |
| mean, | (10) | 16 | 0.27 | 8 | 5 | 0.78 | 433 | 5 | 1.08 | 734 | 6 | 0.84 | 435 |
| max/fails | | | 0/0 | | | 0/0 | | | 0/0 | | | 0/0 | |
| Ferrier | 5-100 | 5 | 0.88 | 14 | 14 | 9.07 | 4657 | 14 | 15.64 | 17958 | 14 | 9.16 | 4661 |
| | 6-100 | 4 | 12.20 | 114 | 5 | 2.58 | 1748 | 5 | 3.18 | 2916 | 5 | 2.62 | 1752 |
| mean, | (20) | 4 | 6.54 | 64 | 10 | 5.83 | 3203 | 10 | 9.41 | 10437 | 10 | 5.89 | 3207 |
| max/fails | | | 1/1 | | | 0/2 | | | 0/0 | | | 0/0 | |
| | 10-100 | 3 | 0.54 | 13 | 16 | 22.09 | 4676 | 16 | 112.32 | 43976 | 16 | 22.17 | 4679 |
| | 11-100 | 3 | 5.44 | 113 | 16 | 17.69 | 3676 | 16 | 18.38 | 3978 | 16 | 17.81 | 3680 |
| | 12-100 | 4 | 5.65 | 121 | 4 | 29.24 | 5882 | 16 | 40.52 | 10822 | 5 | 29.49 | 5887 |
| | 13-500 | $\neq$ stationary points | | | | | | | | | | | |
| LV | 14-500 | 4 | 4.25 | 13 | 16 | 338.32 | 21189 | 16 | 757.08 | 60490 | 16 | 338.82 | 21191 |
| | 15-500 | $\neq$ stationary points | | | | | | | | | | | |
| | 16-500 | $\neq$ stationary points | | | | | | | | | | | |
| | 9-100 | $\neq$ stationary points | | | | | | | | | | | |
| mean, | (40) | 4 | 3.97 | 65 | 13 | 101.84 | 8856 | 16 | 232.08 | 29816 | 13 | 102.08 | 8860 |
| max/fails | | | 0/0 | | | 1/5 | | | 0/4 | | | 0/0 | |
| | 6-100 | 2 | 90.94 | 105 | 4 | 165.89 | 6536 | 16 | 804.07 | 42648 | 5 | 172.30 | 6544 |
| MQ | 7-100 | 16 | 0.15 | 9 | 5 | 12.16 | 5122 | 5 | 13.23 | 6178 | 6 | 12.27 | 5125 |
| | 8-500 | -0 | 5023.56 | 22 | 0 | 14634.68 | 8352 | 0 | 75735.20 | 42086 | 16 | 20200.74 | 8384 |
| | 9-500 | 0 | 478.97 | 25 | 3 | 991.68 | 6572 | 16 | 5382.02 | 42461 | 3 | 1016.28 | 6575 |
| mean, | (40) | 4 | 1398.41 | 40 | 3 | 3951.10 | 6646 | 9 | 20483.63 | 33343 | 8 | 5350.40 | 6657 |
| max/fails | | | 6/6 | | | 6/8 | | | 0/5 | | | 2/2 | |
| | 13-100 | 16 | 1.35 | 23 | 5 | 0.63 | 410 | 5 | 0.80 | 711 | 5 | 0.80 | 413 |
| | 14-100 | 4 | 7.81 | 36 | 16 | 13.09 | 4443 | 16 | 58.45 | 43744 | 16 | 13.41 | 4447 |
| | 15-100 | 5 | 0.91 | 9 | 16 | 16.35 | 4752 | 16 | 79.97 | 44052 | 16 | 16.86 | 4756 |
| | 16-100 | 12 | 0.88 | 21 | 8 | 0.13 | 83 | 8 | 0.39 | 384 | 16 | 0.26 | 86 |
| | 17-100 | $\neq$ stationary points | | | | | | | | | | | |
| NK | 18-100 | 10 | 0.21 | 8 | 4 | 0.09 | 69 | 4 | 0.29 | 370 | 16 | 0.31 | 75 |
| | 19-500 | 5 | 19.78 | 39 | 4 | 30.79 | 902 | 4 | 32.57 | 1204 | 16 | 39.20 | 920 |
| | 20-500 | 2 | 11.11 | 7 | 16 | 342.25 | 19620 | 16 | 763.23 | 58920 | 16 | 343.85 | 19621 |
| | 21-500 | 6 | 16.54 | 18 | 16 | 451.70 | 21468 | 16 | 1006.16 | 60769 | 16 | 453.87 | 21470 |
| | 22-500 | 16 | 5.44 | 22 | 8 | 2.25 | 68 | 8 | 6.47 | 489 | 11 | 3.20 | 72 |
| | 23-500 | $\neq$ stationary points | | | | | | | | | | | |
| | 24-500 | 16 | 0.83 | 7 | 5 | 2.16 | 72 | 5 | 4.41 | 373 | 8 | 2.64 | 74 |
| mean, | (100) | 9 | 6.49 | 19 | 10 | 85.94 | 5189 | 10 | 195.28 | 21102 | 14 | 87.44 | 5193 |
| max/fails | | | 0/0 | | | 0/8 | | | 0/8 | | | 0/0 | |
| NesChebRos | 4-100 | 16 | 0.84 | 19 | 1 | 29.63 | 5247 | 1 | 112.88 | 44548 | 1 | 29.73 | 5251 |
| mean, | (10) | 16 | 0.84 | 19 | 1 | 29.63 | 5247 | 1 | 112.88 | 44548 | 1 | 29.73 | 5251 |
| max/fails | | | 0/0 | | | 0/10 | | | 0/10 | | | 0/0 | |
| TiltedMax | 4-100 | 3 | 7.84 | 105 | 7 | 1.59 | 1372 | 7 | 2.04 | 2776 | 7 | 1.62 | 1374 |
| mean, | (10) | 3 | 7.84 | 105 | 7 | 1.59 | 1372 | 7 | 2.04 | 2776 | 7 | 1.62 | 1374 |
| max/fails | | | 2/2 | | | 0/0 | | | 0/0 | | | 0/0 | |
| Ury | 7-100 | 1 | 70.85 | 94 | 7 | 61.95 | 5854 | 7 | 269.76 | 45155 | 16 | 95.00 | 5897 |
| | 8-100 | 0 | 25.19 | 91 | 16 | 61.52 | 5685 | 16 | 277.93 | 44986 | 16 | 88.49 | 5774 |

Continued on next page

| Table 12 – continued from previous page | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (bb) | #-$n$ | CBunf | | | BFGSf | | | Hansof | | | HyCBf | | |
| | | RA | $\overline{\text{sec}}$ | $\overline{\text{(bb)}}$ | RA | $\overline{\text{sec}}$ | $\overline{\text{(bb)}}$ | RA | $\overline{\text{sec}}$ | $\overline{\text{(bb)}}$ | RA | $\overline{\text{sec}}$ | $\overline{\text{(bb)}}$ |
| mean, | (20) | 0 | 48.02 | 92 | 12 | 61.73 | 5770 | 12 | 273.85 | 45070 | 16 | 91.74 | 5836 |
| max/fails | | | 20/20 | | | 14/20 | | | 0/20 | | | 13/13 | |
| MEAN, | (310) | 7 | 163.67 | 47 | 7 | 480.19 | 4295 | 8 | 2379.65 | 21195 | 9 | 639.26 | 4307 |
| max/fails | | | 29/29 | | | 21/53 | | | 0/47 | | | 15/15 | |

Table 12: Results for Group 3: problems in Table 3, $n = 100$ and $n = 500$.