Inexact penalty decomposition methods for optimization problems with geometric constraints

Christian Kanzow[1] · Matteo Lapucci[2]

Abstract

This paper provides a theoretical and numerical investigation of a penalty decomposition scheme for the solution of optimization problems with geometric constraints. In particular, we consider some situations where parts of the constraints are nonconvex and complicated, like cardinality constraints, disjunctive programs, or matrix problems involving rank constraints. By a variable duplication and decomposition strategy, the method presented here explicitly handles these difficult constraints, thus generating iterates which are feasible with respect to them, while the remaining (standard and supposingly simple) constraints are tackled by sequential penalization. Inexact optimization steps are proven sufficient for the resulting algorithm to work, so that it is employable even with difficult objective functions. The current work is therefore a significant generalization of existing papers on penalty decomposition methods. On the other hand, it is related to some recent publications which use an augmented Lagrangian idea to solve optimization problems with geometric constraints. Compared to these methods, the decomposition idea is shown to be numerically superior since it allows much more freedom in the choice of the subproblem solver, and since the number of certain (possibly expensive) projection steps is significantly less. Extensive numerical results on several highly complicated classes of optimization problems in vector and matrix spaces indicate that the current method is indeed very efficient to solve these problems.

✉ Matteo Lapucci
matteo.lapucci@unifi.it

1 Institute of Mathematics, University of Würzburg, Würzburg 97074, Germany

2 Department of Information Engineering, University of Florence, Via di Santa Marta, 3, 50139 Florence, Italy

🖄 Springer

1 Introduction

We consider the program

$$\min_x \ f(x) \quad \text{s.t.} \quad G(x) \in C, \ x \in D, \tag{1.1}$$

where $f : \mathbb{X} \to \mathbb{R}$ and $G : \mathbb{X} \to \mathbb{Y}$ are continuously differentiable mappings, $\mathbb{X}$ and $\mathbb{Y}$ are Euclidean spaces, i.e., real and finite-dimensional Hilbert spaces, $C \subseteq \mathbb{Y}$ is nonempty, closed, and convex, whereas $D \subseteq \mathbb{X}$ is only assumed to be nonempty and closed (not necessarily convex), representing a possibly complicated set, for which, however, a projection operation is accessible.

This very general setting (analyzed for example in [1]) covers, for example, standard nonlinear programming problems with convex constraints, but also difficult *disjunctive programming* problems [2–5], e.g., complementarity [6], vanishing [7], switching [8] and cardinality constrained [9, 10] problems. Matrix optimization problems such as low-rank approximation [11, 12] are also captured by our setting.

In recent years, lots of studies have been published that deal with problems with this structure, where the feasible set consists of the intersection of a collection of analytical constraints and a complicated, irregular set, manageable, for example, by easy projections. In particular, approaches based on decomposition and sequential penalty or augmented Lagrangian methods have been proposed for the convex case [13], the cardinality constrained case [10, 14] and the low-rank approximation case [15]; the recurrent idea in all these works consists of the application of the variable splitting technique [16, 17], to then define a penalty function associated with the differentiable constraints and the additional equality constraint linking the two blocks of variables and finally solve the problem by a sequential penalty method. The optimization of the penalty function is carried out by a two-block alternating minimization scheme [18], which can be run in an exact [14, 15] or inexact [10, 13] fashion.

The aim of this work is to extend the *inexact Penalty Decomposition* approach to the general setting (1.1) in such a way that it can deal with arbitrary abstract constraints D (at least theoretically, in practice D needs to be such that projections onto this set are easy to compute) and that it allows additional (seemingly simple) constraints given by $G(x) \in C$. This setting is related to some recent work on (safeguarded) augmented Lagrangian methods, see, in particular, [1], where the resulting subprobems are solved by a projected gradient-type method, which might be inefficient especially for ill-conditioned problems. The decomposition idea used here allows a much wider choice of subproblem solvers, usually resulting in a more efficient solver of the given optimization problem (1.1).

The paper is organized as follows: Sect. 2 summarizes some preliminary concepts and results. In particular, we recall the definitions of an M-stationary point (the counterpart of a KKT point for the general setting from (1.1)), of an AM-stationary point (as a sequential version of M-stationarity) and of an AM-regular point (this being a suitable and relatively weak constraint qualification). Section 3 then presents the *Penalty Decomposition (PD)* method together with a global convergence theory, assuming that the resulting subproblems can be solved inexactly up to a certain degree. In Sect. 4, we

then present a class of *inexact alternating minimization* methods [18, 19] which, under certain assumptions, are guaranteed to find the desired approximate solution of the subproblems arising in the outer penalty scheme. The remaining part of the paper is then devoted to the implementation of the overall method and corresponding numerical results. To this end, Sect. 5 first discusses several instances of the general setting (1.1) with difficult constraints D where our method can be applied to quite efficiently since projections onto D are simple and/or known analytically (though the latter does not necessarily imply that these projections are easy to compute numerically). In Sect. 6, we then present the results of an extensive numerical testing, where we also compare our method, using different realizations, with the augmented Lagrangian method from [1]. We conclude with some final remarks in Sect. 7.

2 Preliminaries

The Euclidean projection $P_C : \mathbb{Y} \to \mathbb{Y}$ onto the nonempty, closed, and convex set C is defined by
$$P_C(y) := \operatorname*{argmin}_{z \in C} \|z - y\|.$$

The corresponding distance function $d_C : \mathbb{Y} \to \mathbb{R}$ can then be written as

$$\operatorname{dist}_C(y) := \min_{z \in C} \|z - y\| = \|P_C(y) - y\|.$$

Note that the distance function is nonsmooth (in general), but the squared distance function
$$s_C(y) := \frac{1}{2}\operatorname{dist}_C^2(y)$$

is continuously differentiable everywhere with derivative given by

$$\nabla s_C(y) = y - P_C(y), \tag{2.1}$$

see [20, Cor. 12.30]. Moreover, projections onto the nonempty and closed set D also exist, but are not necessarily unique. Therefore, we define the (usually set-valued) projection operator $\Pi_D : \mathbb{X} \rightrightarrows \mathbb{X}$ by

$$\Pi_D(x) := \operatorname*{argmin}_{z \in D} \|z - x\| \neq \emptyset.$$

The corresponding distance function $\operatorname{dist}_D(\cdot)$ is, of course, single-valued again. Furthermore, given a set-valued mapping $S : \mathbb{X} \rightrightarrows \mathbb{X}$ on an arbitrary Euclidean space $\mathbb{X}$, we define the *outer limit* of S at a point $\bar{x}$ by

$$\limsup_{x \to \bar{x}} S(x) := \left\{ y \in \mathbb{X} \mid \exists x^k \to \bar{x}, \exists y^k \to y \text{ with } y^k \in S(x^k) \ \forall k \in \mathbb{N} \right\}.$$

This allows to define the *limiting normal cone* at a point $x \in D$ by

$$\mathcal{N}_D^{\lim}(x) := \limsup_{v \to x} \big(\mathrm{cone}\big(v - \Pi_D(v)\big)\big),$$

see [21, Sect. 1.1] for further details. Writing

$$x \to_D \bar{x} \iff x \to \bar{x}, \ x \in D$$

for sequences converging to an element $\bar{x} \in D$ such that the whole sequence belongs to D, the limiting normal cone has the important robustness property

$$\limsup_{x \to_D \bar{x}} \mathcal{N}_D^{\lim}(x) = \mathcal{N}_D^{\lim}(\bar{x}) \tag{2.2}$$

that will be exploited heavily in our subsequent analysis, see [21, Prop. 1.3].

Note that, for D being convex, this limiting normal cone reduces to the standard normal cone from convex analysis, i.e., we have

$$\mathcal{N}_D^{\lim}(\bar{x}) = \mathcal{N}_D(\bar{x}) := \{\lambda \in \mathbb{X} \mid \langle \lambda, x - \bar{x} \rangle \le 0 \ \forall x \in D \}$$

for any given $\bar{x} \in D$. For points $\bar{x} \notin D$, we set $\mathcal{N}_D^{\lim}(\bar{x}) := \mathcal{N}_D(\bar{x}) := \emptyset$. For the convex set C, the standard normal cone and the projection operator are related by

$$p = P_C(y) \iff y - p \in \mathcal{N}_C(p), \tag{2.3}$$

see [20, Prop. 6.46].

We next introduce a stationarity condition which generalizes the concept of a KKT point to constrained optimization problems with possibly difficult constraints as given by the set D in our setting (1.1), see [22] for a general discussion and, e.g., [1] for a realization of the specific setting considered here.

Definition 2.1 A feasible point $\bar{x} \in \mathbb{X}$ of the optimization problem (1.1) is called an *M-stationary point* (Mordukhovich-stationary point) of (1.1) if there exists a multiplier $\lambda \in \mathbb{Y}$ such that

$$0 \in \nabla f(\bar{x}) + G'(\bar{x})^*\lambda + \mathcal{N}_D^{\lim}(\bar{x}), \quad \lambda \in \mathcal{N}_C\big(G(\bar{x})\big).$$

Note that this definition coincides with the one of a KKT point if D is convex. The following is a sequential version of M-stationarity.

Definition 2.2 A feasible point $\bar{x} \in \mathbb{X}$ of the optimization problem (1.1) is called an *AM-stationary point* (asymptotically M-stationary point) of (1.1) if there exist sequences $\{x^k\}, \{\varepsilon^k\} \subseteq \mathbb{X}$ and $\{\lambda^k\}, \{z^k\} \subseteq \mathbb{Y}$ such that $x^k \to \bar{x}, \varepsilon^k \to 0, z^k \to 0$, as well as

$$\varepsilon^k \in \nabla f(x^k) + G'(x^k)^*\lambda^k + \mathcal{N}_D^{\lim}(x^k), \quad \lambda^k \in \mathcal{N}_C\big(G(x^k) - z^k\big)$$

for all $k \in \mathbb{N}$.

We stress that this definition implicitly includes that the sequences $\{x^k\}$ and $\{G(x^k) - z^k\}$ belong to the sets D and C, respectively, since otherwise the corresponding normal cones would be empty. We also note that the previous definition generalizes the related concept of AKKT points introduced for standard nonlinear programs in [23] to our setting with the more difficult constraints. In a similar way, the subsequent regularity conditions are also motivated by related ones from [24], where they were presented for standard nonlinear programs. Their generalizations to our setting can be found in [1, 22], for example.

Every M-stationary point is obviously also AM-stationary, whereas the opposite implication will be guaranteed to hold by a regularity condition that will now be introduced. To this end, let us write

$$\mathcal{M}(x, z) := G'(x)^* \mathcal{N}_C(G(x) - z) + \mathcal{N}_D^{\lim}(x).$$

Recall that $\mathcal{N}_D^{\lim}(x)$ is nonempty if and only if $x \in D$, which is therefore an implicit requirement for the set $\mathcal{M}(x, z)$ to be nonempty. Moreover, we consider the set

$$\limsup_{x \to \bar{x}, z \to 0} \mathcal{M}(x, z) = \left\{ v \mid \exists x^k \to_D \bar{x}, \exists z^k \to 0 : v^k \to v \text{ and } v^k \in \mathcal{M}(x^k, z^k) \ \forall k \in \mathbb{N} \right\}.$$

Note that the auxiliary sequence $\{z^k\}$ needs to be introduced since the elements $G(x^k)$ do not necessarily belong to C, whereas x^k is supposed to be an element of D.

Definition 2.3 Let $\bar{x}$ be feasible for (1.1). Then $\bar{x}$ is called *AM-regular* for (1.1) if
$$\limsup_{x \to \bar{x}, z \to 0} \mathcal{M}(x, z) \subseteq \mathcal{M}(\bar{x}, 0).$$

Using this terminology, the following statements hold, cf. [22] for further details.

Theorem 2.4 *The following statements hold:*

(a) *Every local minimum of* (1.1) *is an AM-stationary point.*
(b) *If $\bar{x}$ is an AM-stationary point satisfying AM-regularity, then $\bar{x}$ is an M-stationary point of* (1.1).
(c) *Conversely, if for every continuously differentiable function f, the implication*

$$\bar{x} \text{ is an AM-stationary point} \implies \bar{x} \text{ is an M-stationary point}$$

holds for the corresponding optimization problem (1.1), *then $\bar{x}$ is AM-regular.*

Statement (a) shows that every local minimum of (1.1) is an AM-stationary point even in the absense of any constraint qualification (CQ for short). Hence AM-stationary is a (sequential) first-order optimality condition. In order to guarantee that an AM-stationary point is already an M-stationary point (hence a KKT point in the standard setting of a nonlinear program, say), we require a CQ, namely the AM-regularity condition, cf. Theorem 2.4 (b). The final statement (c) of that result shows that, in a certain sense, AM-regularity is the weakest CQ which implies AM-stationary points to be M-stationary. In fact, this AM-regularity condition turns out to be a fairly weak

condition. For example, for standard nonlinear programs, AM-regularity is stronger than the Abadie CQ, but weaker than most of the other well-known CQs like MFCQ (Mangasarian-Fromovitz CQ), CRCQ (constant rank CQ), CPLD (constant positive linear dependence), and RCPLD (relaxed CPLD), to mention at least some of the more prominent ones. We refer the interested reader to [22, 25] and references therein for further details.

The algorithm, to be described in the following section, is based on the reformulation

$$\min_{x,y} f(x) \quad \text{s.t.} \quad x - y = 0, \ G(x) \in C, y \in D, \tag{2.4}$$

of the given optimization problem (1.1). The previous notions of M- and AM-stationarity and AM-regularity can be directly translated to this program by observing that (2.4) can be written in the format of (1.1) as

$$
\begin{aligned}
\tilde{x} &:= (x, y), \\
\tilde{f}(\tilde{x}) &:= \tilde{f}(x, y) = f(x), \\
\tilde{G}(\tilde{x}) &:= \tilde{G}(x, y) = \begin{pmatrix} G(x) \\ x - y \end{pmatrix}, \\
\tilde{C} &:= C \times 0_{\mathbb{X}}, \\
\tilde{D} &:= \mathbb{X} \times D.
\end{aligned}
$$

The counterpart of Theorem 2.4 then also holds for the corresponding (asymptotic) M-stationarity and regularity concepts defined for the formulation (2.4). Note that regularity conditions and constraint qualifications depend on the explicit formulation of a constraint system, hence the corresponding concepts are not necessarily equivalent for the two formulations of our program. We stress, however, that the important notion of an M-stationary point for (2.4) is equivalent to the notion of an M-stationary point for (1.1).

For the sake of completeness, and since this condition will be used explicitly in our convergence analysis, let us write down explicitly the resulting AM-stationarity condition for the reformulated program (2.4): with the above identifications, a feasible point $\tilde{x}^*$ of (2.4) is AM-stationary if there exist sequences $\{\tilde{x}^k\}$, $\{\tilde{\varepsilon}^k\}$, and $\{\tilde{\lambda}^k\}$, $\{\tilde{z}^k\}$ such that $\tilde{x}^k \to \tilde{x}^*$, $\tilde{z}^k \to 0$, $\tilde{\varepsilon}^k \to 0$ as well as

$$\tilde{\varepsilon}^k \in \nabla \tilde{f}(\tilde{x}^k) + \tilde{G}'(\tilde{x}^k)^* \tilde{\lambda}^k + \mathcal{N}_{\tilde{D}}^{\lim}(\tilde{x}^k) \quad \text{and} \quad \tilde{\lambda}^k \in \mathcal{N}_{\tilde{C}}\big(\tilde{G}(\tilde{x}^k) - \tilde{z}^k\big) \tag{2.5}$$

for all k. Using the definitions of $\tilde{f}, \tilde{G}$ etc., exploiting standard properties of the limiting and standard normal cones (in particular, the Cartesian product rule, cf. [26, Prop. 6.41]), and writing $\tilde{x}^k =: (x^k, y^k)$, $\tilde{\lambda}^k =: (\lambda^k, \mu^k)$ as well as z^k for the first block of $\tilde{z}^k$ (the second block component of $\tilde{z}^k$ turns out to be irrelevant), we see that the two conditions from (2.5) can be rewritten as

$$\tilde{\varepsilon}^k \in \begin{pmatrix} \nabla f(x^k) + G'(x^k)^* \lambda^k + \mu^k \\ -\mu^k + \mathcal{N}_D^{\lim}(y^k) \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \lambda^k \\ \mu^k \end{pmatrix} \in \begin{pmatrix} \mathcal{N}_C\big(G(x^k) - z^k\big) \\ \mathbb{X} \end{pmatrix}. \tag{2.6}$$

3 Algorithm and convergence

The algorithm to be presented here is based on the reformulation (2.4) of the given program (1.1). The idea is to take advantage of the fact that the constraints $G(x) \in C$ and $y \in D$ occur in a decomposed way. This formulation allows to develop an alternating direction-type penalty scheme for the solution of the orginal problem (1.1). To this end, let $\tau > 0$ be a penalty parameter and define the partial penalty function

$$q_\tau(x, y) := f(x) + \frac{\tau}{2}\Big(\|x - y\|^2 + \mathrm{dist}_C^2\big(G(x)\big)\Big). \tag{3.1}$$

Note that q_τ does not include the potentially difficult constraint $x \in D$, which we therefore have to deal with explicitly. The general algorithmic scheme that we will investigate here is summarized in Algorithm 1.

Algorithm 1 Inexact penalty decomposition method

(S.0) Choose $\delta_0 \geq 0$ and $\tau_0 > 0$, a starting point $(x^0, y^0) \in \mathbb{X} \times \mathbb{X}$, and set $k := 0$.
(S.1) If a suitable termination criterion holds: STOP.
(S.2) Compute $\big(x^{k+1}, y^{k+1}\big)$ such that

$$\big\|\nabla_x q_{\tau_k}(x^{k+1}, y^{k+1})\big\| \leq \delta_k \tag{3.2}$$

and

$$y^{k+1} \in \mathrm{argmin}_{y \in D}\, q_{\tau_k}(x^{k+1}, y) \tag{3.3}$$

hold.
(S.3) Choose $\delta_{k+1} \leq \delta_k$, $\tau_{k+1} > \tau_k$, $k \leftarrow k + 1$, and go to (S.1).

Note that Algorithm 1 is a very general scheme for the solution of the reformulated problem (2.4). The main computational burden is in step (S.2). We will see how this step can be realized by an alternating minimization-type iteration in Sect. 4. Here we only note that the computation of the exact minimizer $y^{k+1} = \mathrm{argmin}_{y \in D}\, q_{\tau_k}(x^{k+1}, y)$ can be carried out very easily if projections onto the set D can be computed efficiently (we refer the reader to Sects. 5, 6 for some examples). This follows immediately from the definition of q_{τ_k}, which implies that y^{k+1} is characterized by

$$y^{k+1} \in \Pi_D(x^{k+1}). \tag{3.4}$$

The remaining part of this section is devoted to the global convergence properties of the general scheme from Algorithm 1. The technique of proof patterns the one used in [1] for an augmented Lagrangian method.

We begin with a feasibility-type result. To this end, recall that all penalty-type methods suffer from the fact that accumulation points may not be feasible for the given optimization problem. The following result shows that such an accumulation point still has a very nice property.

Proposition 3.1 *Let $\{(x^k, y^k)\}$ be a sequence generated by Algorithm 1 with $\{\delta_k\}$ being bounded and $\{\tau_k\} \to \infty$. Then every accumulation point $(\bar{x}, \bar{y})$ of the sequence $\{(x^k, y^k)\}$ is an M-stationary point of the feasibility problem*

$$\min_{x,y} \ \frac{1}{2} dist_C^2(G(x)) + \frac{1}{2}\|x - y\|^2 \quad s.t. \quad y \in D. \tag{3.5}$$

Proof Let $\{(x^{k+1}, y^{k+1})\}_K$ be a subsequence converging to $(\bar{x}, \bar{y})$. Using the derivative formula of the distance function from (2.1) together with the chain rule, we have, by construction,

$$\begin{aligned}
&\left\| \nabla_x q_{\tau_k}(x^{k+1}, y^{k+1}) \right\| \\
&= \left\| \nabla f(x^{k+1}) + \tau_k \left[G'(x^{k+1})^* \left[G(x^{k+1}) - P_C\big(G(x^{k+1})\big) \right] + x^{k+1} - y^{k+1} \right] \right\| \\
&\leq \delta_k
\end{aligned}$$

and

$$0 \in \nabla_y q_{\tau_k}(x^{k+1}, y^{k+1}) + \mathcal{N}_D^{\lim}(y^{k+1}) = \tau_k(y^{k+1} - x^{k+1}) + \mathcal{N}_D^{\lim}(y^{k+1})$$

for all $k \in \mathbb{N}$. Dividing the first equation by τ_k and exploiting the cone property in the second inclusion yields

$$\left\| \frac{1}{\tau_k} \nabla f(x^{k+1}) + G'(x^{k+1})^* \left[G(x^{k+1}) - P_C\big(G(x^{k+1})\big) \right] + x^{k+1} - y^{k+1} \right\| \leq \frac{\delta_k}{\tau_k}$$

and

$$0 \in y^{k+1} - x^{k+1} + \mathcal{N}_D^{\lim}(y^{k+1})$$

for all $k \in \mathbb{N}$, cf. [26, Thm. 6.12]. Taking the limit $k \to_K \infty$, using the continuity of $\nabla f, G, G', P_C$, and the robustness property (2.2) of the limiting normal cone, we obtain

$$G'(\bar{x})^*\big(G(\bar{x}) - P_C\big(G(\bar{x})\big)\big) + \bar{x} - \bar{y} = 0 \quad \text{and} \quad 0 \in \bar{y} - \bar{x} + \mathcal{N}_D^{\lim}(\bar{y}).$$

This shows that $(\bar{x}, \bar{y})$ is an M-stationary point of (3.5). $\qquad\square$

Recall that Algorithm 1 automatically generates iterates y^k which belong to the set D. The objective function in (3.5) therefore only measures the violation of the constraints $G(x) \in C$ and $x - y = 0$, which is included in the penalty term of q_τ, i.e., (3.5) is a feasibility problem of the decomposed problem (2.4). If $\bar{x} = \bar{y}$, then $\bar{x}$ turns out to be an M-stationary point of

$$\min_x \ \frac{1}{2} dist_C^2(G(x)) \quad s.t. \quad x \in D,$$

which is the feasibility problem of the original problem (1.1). Though Proposition 3.1 obviously does not guarantee that an accumulation point is feasible (either for the

original or the decomposed formulation), it guarantees at least a stationarity property, which is the best one can expect in general. Moreover, if the feasible set of (1.1) is nonempty and the function $\frac{1}{2}\text{dist}_C^2\big(G(x)\big)$ of (3.5) is convex, then every M-stationary point is a global minimum and, hence, a feasible point of (1.1) or (2.4). Note that the square of the above distance function is automatically convex if the constraint $G(x) \in C$ satisfies standard conditions which imply that this set is convex by itself.

Moreover, one can also define an extended Robinson-type constraint qualification (which boils down to the extended MFCQ condition for standard nonlinear programs) which automatically imply that accumulation points of a sequence generated by Algorithm 1 are feasible, cf. [27, 28] for further details.

Hence, under reasonable assumptions, we can guarantee that accumulation points are automatically feasible for (1.1) or (2.4), whereas, in general, they are at least M-stationary points for the feasibility problem. The following global convergence result therefore assumes that we have a feasible accumulation point and shows that this one is automatically AM-stationary for problem (2.4).

Theorem 3.2 *Let $\{(x^k, y^k)\}$ be a sequence generated by Algorithm 1 with $\{\delta_k\} \to 0$ and $\{\tau_k\} \to \infty$, and let $(\bar{x}, \bar{y})$ be an accumulation point of this sequence that is feasible for (2.4). Then $(\bar{x}, \bar{y})$ is an AM-stationary point of the optimization problem (2.4).*

Proof Let $\{(x^{k+1}, y^{k+1})\}_K$ be a subsequence converging to $(\bar{x}, \bar{y})$. Recall that $(\bar{x}, \bar{y})$ is feasible for (2.4), hence $G(\bar{x}) \in C$ and $\bar{x} = \bar{y} \in D$. We further define the sequences

$$
\begin{aligned}
z^{k+1} &:= G(x^{k+1}) - P_C\big(G(x^{k+1})\big), \\
\lambda^{k+1} &:= \tau_k\big(G(x^{k+1}) - P_C\big(G(x^{k+1})\big)\big), \\
\mu^{k+1} &:= \tau_k\big(x^{k+1} - y^{k+1}\big), \\
\varepsilon^{k+1} &:= \nabla f(x^{k+1}) + \tau_k\big[G'(x^{k+1})^*\big(G(x^{k+1}) - P_C(G(x^{k+1}))\big) + x^{k+1} - y^{k+1}\big].
\end{aligned}
$$

Then setting

$$
\tilde{x}^{k+1} := \begin{pmatrix} x^{k+1} \\ y^{k+1} \end{pmatrix}, \quad \tilde{\varepsilon}^{k+1} := \begin{pmatrix} \varepsilon^{k+1} \\ 0 \end{pmatrix}, \quad \tilde{\lambda}^{k+1} := \begin{pmatrix} \lambda^{k+1} \\ \mu^{k+1} \end{pmatrix}, \quad \tilde{z}^{k+1} := \begin{pmatrix} z^{k+1} \\ 0 \end{pmatrix},
$$

we claim that the corresponding four (sub-) sequences $\{\tilde{x}^{k+1}\}_K = \{(x^{k+1}, y^{k+1})\}_K$, $\{\tilde{z}^{k+1}\} = \{(z^{k+1}, 0)\}_K$, $\{\tilde{\varepsilon}^{k+1}\}_K = \{(\varepsilon^{k+1}, 0)\}_K$, and $\{\tilde{\lambda}^{k+1}\}_K = \{(\lambda^{k+1}, \mu^{k+1})\}_K$ satisfy the properties of an AM-stationary point for problem (2.4) as stated at the end of Sect. 2, cf. (2.5) and (2.6). First of all, $(\bar{x}, \bar{y})$ is feasible and $\{(x^{k+1}, y^{k+1})\}_K \to (\bar{x}, \bar{y})$ by assumption. Furthermore, by definition of ε^{k+1} and the construction of Algorithm 1, we also have

$$
\|\varepsilon^{k+1}\| = \|\nabla_x q_{\tau_k}(x^{k+1}, y^{k+1})\| \le \delta_k \to 0.
$$

This obviously implies $\|\tilde{\varepsilon}^{k+1}\| \to 0$. Furthermore, the definitions of λ^{k+1} and μ^{k+1} together with $0 \in \tau_k\big(y^{k+1} - x^{k+1}\big) + \mathcal{N}_D^{\lim}(y^{k+1})$ yield

$$
\varepsilon^{k+1} = \nabla f(x^{k+1}) + G'(x^{k+1})^*\lambda^{k+1} + \mu^{k+1} \quad \text{and} \quad 0 \in -\mu^{k+1} + \mathcal{N}_D^{\lim}(y^{k+1}),
$$

hence the first inclusion in (2.6) holds. To verify the second inclusion, we only have to take a closer look at the first block. By definition of λ^{k+1} and the relation (2.3) between the projection and the normal cone, we get

$$\lambda^{k+1} = \tau_k\big(G(x^{k+1}) - P_C\big(G(x^{k+1})\big)\big) \in \mathcal{N}_C\big(P_C(G(x^{k+1}))\big) = \mathcal{N}_C\big(G(x^{k+1}) - z^{k+1}\big),$$

where the last identity comes from the definition of z^{k+1}. Finally, we also have $\bar{z}^{k+1} \to_K 0$ since z^{k+1} satisfies

$$z^{k+1} = G(x^{k+1}) - P_C\big(G(x^{k+1})\big) \to_K G(\bar{x}) - P_C\big(G(\bar{x})\big) = 0$$

by the continuity of G and the projection operator P_C as well as the feasibility of $\bar{x}$. Altogether, this shows that $(\bar{x}, \bar{y})$ is an AM-stationary point of the program (2.4). □

Using Theorem 3.2 together with the counterpart of Theorem 2.4 for (2.4) and the fact that the M-stationarity conditions for the two problems (1.1) and (2.4) are equivalent, we directly obtain the following result.

Theorem 3.3 *Let $\{(x^k, y^k)\}$ be a sequence generated by Algorithm 1 with $\{\delta_k\} \to 0$ and $\{\tau_k\} \to \infty$, and let $(\bar{x}, \bar{y})$ be an accumulation point of this sequence that is feasible and satisfies AM-regularity for (2.4). Then $(\bar{x}, \bar{y})$ is an M-stationary point of the optimization problem (2.4), and $\bar{x}$ itself is an M-stationary point of the original problem (1.1).*

4 Solution of subproblems by inexact alternating minimization

The Penalty Decomposition approach basically consists in approximately solving the sequence of penalty subproblems at step (S.2) by a two-block decomposition method. The alternating minimization loop can be stopped, at each iteration, as soon as an approximate stationary point of the penalty function w.r.t. the first block of variables x is attained. The instructions of the (inexact) Alternating Minimization loop at a fixed iteration k of the Penalty Decomposition method are detailed in Algorithm 2.

Algorithm 2 Inexact Alternating Minimization

(S.0) Given $\delta_k \geq 0$ and $\tau_k > 0$, a starting point $(x^k, y^k) \in \mathbb{X} \times D$, $\gamma \in (0, 1)$, $\beta \in (0, 1)$, set $\ell := 0$, $(u^0, v^0) = (x^k, y^k)$.

(S.1) If $\big\|\nabla_x q_{\tau_k}(u^\ell, v^\ell)\big\| \leq \delta_k$: STOP returning $(x^{k+1}, y^{k+1}) = (u^\ell, v^\ell)$.

(S.2) Choose a positive definite self-adjoint linear map H_ℓ, set $d^\ell = -H_\ell(\nabla_x q_{\tau_k}(u^\ell, v^\ell))$ and compute

$$\alpha_\ell = \max_{j \in \mathbb{N}}\{\beta^j : q_{\tau_k}(u^\ell + \beta^j d^\ell, v^\ell) \leq q_{\tau_k}(u^\ell, v^\ell) + \gamma\beta^j \langle\nabla_x q_{\tau_k}(u^\ell, v^\ell), d^\ell\rangle\} \qquad (4.1)$$

(S.3) Set $u^{\ell+1} = u^\ell + \alpha_\ell d_\ell$.

(S.4) Compute $v^{\ell+1} \in \underset{v \in D}{\mathrm{argmin}}\, q_{\tau_k}(u^{\ell+1}, v) = \Pi_D(u^{\ell+1})$.

(S.5) Set $\ell = \ell + 1$ and go to (S.1).

As already pointed out, if we assume that projections onto the set D are easily computable, the update of the second block of variables can be carried out exactly by (3.4).

On the other hand, an exact x-update step (i.e., finding a global minimizer of $q_{\tau_k}(u, v^\ell)$) may be prohibitive in most applications. For this reason, the x-variable is only updated by a descent step along a descent direction, with a step size selected by an Armijo-type line search.

Note that the direction $d_\ell = -H_\ell(\nabla_x q_{\tau_k}(u^\ell, v^\ell))$ is certainly a descent direction, since H_ℓ is positive definite, $\nabla_x q_{\tau_k}(u^\ell, v^\ell) \neq 0$ and, thus,

$$\langle \nabla_x q_{\tau_k}(u^\ell, v^\ell), d^\ell \rangle = -\langle \nabla_x q_{\tau_k}(u^\ell, v^\ell), H_\ell(\nabla_x q_{\tau_k}(u^\ell, v^\ell)) \rangle < 0. \qquad (4.2)$$

The Armijo line search provides a sufficient decrease granting, under suitable assumptions on the sequence of maps H_ℓ, the convergence of the entire alternate minimization scheme.

Note that by properly choosing H_ℓ we can retrieve the descent directions employed in most widely employed nonlinear optimization solvers. This point, which we will emphasize again later on, is particularly relevant from the computational point of view.

Throughout this section, we make the following assumption.

Assumption 4.1 $f(x)$ has bounded level sets upon $\mathbb{X}$, i.e., $\mathcal{L}_f(\eta) = \{x \in \mathbb{X} \mid f(x) \leq \eta\}$ is bounded for any $\eta \in \mathbb{R}$.

We then begin by proving that, under Assumption 4.1, the penalty function has bounded level sets for any nonnegative value of the penalty parameter τ.

Lemma 4.2 *The penalty function $q_\tau(x, y)$ has bounded level sets for any $\tau \geq 0$.*

Proof Consider any $\eta \in \mathbb{R}$. From Assumption 4.1, the level set $\mathcal{L}_f(\eta)$ is bounded. Let us consider $\mathcal{L}_{q_\tau}(\eta)$ for any $\tau \geq 0$.

Assume by contradiction that $\mathcal{L}_{q_\tau}(\eta)$ is not bounded, i.e., there exists $\{(x^t, y^t)\}$ such that $(x^t, y^t) \in \mathcal{L}_{q_\tau}(\eta)$ for all t and $\|(x^t, y^t)\| \to \infty$. Then, either $\|x^t\| \to \infty$ or $\|y^t\| \to \infty$.

If $\|x^t\| \to \infty$, we have $f(x^t) > \eta$ for t sufficiently large, being $\mathcal{L}_f(\eta)$ bounded. But then, from the definition of $q_\tau(x, y)$, we have for t sufficiently large $q_\tau(x^t, y^t) \geq f(x^t) > \eta$, which contradicts $\{(x^t, y^t)\} \subseteq \mathcal{L}_{q_\tau}(\eta)$.

Thus, $\|y^t\| \to \infty$ while $\|x^t\|$ stays bounded. However,

$$q_\tau(x^t, y^t) = f(x^t) + \frac{\tau}{2}\left(\|x^t - y^t\|^2 + \text{dist}_C^2\big(G(x)\big)\right) > \eta$$

for t sufficiently large, as $\|x^t - y^t\|^2 \to \infty$, $\text{dist}_C^2\big(G(x)\big) \geq 0$ and f is bounded having compact level sets. This again is a contradiction, which completes the proof. □

It can be seen that step (S.2) of Algorithm 2 is well-defined, i.e., there exists a finite integer j such that β^j satisfies the acceptability condition (4.1). Moreover the following result can be readily obtained by standard results in nonlinear optimization [29].

Lemma 4.3 *Let $\{(u^\ell, v^\ell)\}$ be the sequence generated by Algorithm 2. Let $T \subseteq \{0, 1, 2, \ldots\}$ be an infinite subset such that*

$$\lim_{\substack{\ell \to \infty \\ \ell \in T}} (u^\ell, v^\ell) = (\bar{u}, \bar{v}).$$

Let $\{d^\ell\}$ be a sequence of directions such that $\langle \nabla_x q_{\tau_k}(u^\ell, v^\ell), d^\ell \rangle < 0$ and assume that $\|d^\ell\| \leq M$ for some $M > 0$ and for all $\ell \in T$. If, for any fixed (outer iteration) k, the following equation holds

$$\lim_{\substack{\ell \to \infty \\ \ell \in T}} q_{\tau_k}(u^\ell, v^\ell) - q_{\tau_k}(u^\ell + \alpha_\ell d^\ell, v^\ell) = 0,$$

then we have

$$\lim_{\substack{\ell \to \infty \\ \ell \in T}} \langle \nabla_x q_{\tau_k}(u^\ell, v^\ell), d^\ell \rangle = 0.$$

Proof Since, for any ℓ, α_ℓ is chosen according to (4.1), we have

$$q_{\tau_k}(u^{\ell+1}, v^\ell) \leq q_{\tau_k}(u^\ell, v^\ell) + \gamma \alpha_\ell \langle \nabla_x q_{\tau_k}(u^\ell, v^\ell), d^\ell \rangle.$$

Taking the limits for $\ell \in T$, $\ell \to \infty$, we get

$$\lim_{\substack{\ell \to \infty \\ \ell \in T}} q_{\tau_k}(u^\ell + \alpha_\ell d^\ell, v^\ell) - q_{\tau_k}(u^\ell, v^\ell) \leq \lim_{\substack{\ell \to \infty \\ \ell \in T}} \gamma \alpha_\ell \langle \nabla_x q_{\tau_k}(u^\ell, v^\ell), d^\ell \rangle \leq 0,$$

where the last inequality comes from the fact that $\gamma > 0$, $\alpha_\ell \geq 0$ and $\langle \nabla_x q_{\tau_k}(u^\ell, v^\ell), d^\ell \rangle < 0$ by assumption. From the hypotheses, we also have that the leftmost limit goes to 0, hence we obtain

$$\lim_{\substack{\ell \to \infty \\ \ell \in T}} \gamma \alpha_\ell \langle \nabla_x q_{\tau_k}(u^\ell, v^\ell), d^\ell \rangle = 0. \tag{4.3}$$

Assume, by contradiction, that $\langle \nabla_x q_{\tau_k}(u^\ell, v^\ell), d^\ell \rangle_T$ does not converge to zero. Passing to a subsequence, if necessary, we may assume that $\lim_{\ell \to_T \infty} \langle \nabla_x q_{\tau_k}(u^\ell, v^\ell), d^\ell \rangle = -\nu$ for some number $\nu > 0$. On the other hand, we have $(u^\ell, v^\ell) \to_T (\bar{u}, \bar{v})$ by assumption, and $\{d^\ell\}$ is bounded, so we may also assume that $\{d^\ell\}_T \to \bar{d}$ for some limit point $\bar{d}$. Altogether, we then have

$$\langle \nabla_x q_{\tau_k}(\bar{u}, \bar{v}), \bar{d} \rangle = \lim_{\substack{\ell \to \infty \\ \ell \in T}} \langle \nabla_x q_{\tau_k}(u^\ell, v^\ell), d^\ell \rangle = -\nu < 0.$$

Exploiting (4.3), we see that $\alpha_\ell \to_T 0$ holds. Consequently, for all $\ell \in T$ sufficiently large, we have $\alpha_\ell < \beta^0 = 1$ and thus

$$q_{\tau_k}\left(u^\ell + \frac{\alpha_\ell}{\beta} d^\ell, v^\ell\right) > q_{\tau_k}(u^\ell, v^\ell) + \gamma \frac{\alpha_\ell}{\beta} \langle \nabla_x q_{\tau_k}(u^\ell, v^\ell), d^\ell \rangle.$$

By the mean-value theorem, we can write

$$q_{\tau_k}\left(u^\ell + \frac{\alpha_\ell}{\beta}d^\ell, v^\ell\right) = q_{\tau_k}(u^\ell, v^\ell) + \frac{\alpha_\ell}{\beta}\langle \nabla_x q_{\tau_k}(z^\ell, v^\ell), d^\ell\rangle$$

for some $z^\ell = u^\ell + \theta_\ell \frac{\alpha_\ell}{\beta}d^\ell, \theta_\ell \in (0, 1)$. Subtracting the last two relations and dividing by α_ℓ/β, we get

$$0 > \gamma\langle \nabla_x q_{\tau_k}(u^\ell, v^\ell), d^\ell\rangle - \langle \nabla_x q_{\tau_k}(z^\ell, v^\ell), d^\ell\rangle.$$

On the other hand,

$$\lim_{\substack{\ell \in T \\ \ell \to \infty}} z^\ell = \lim_{\substack{\ell \in T \\ \ell \to \infty}} u^\ell + \theta_\ell \frac{\alpha_\ell}{\beta}d^\ell = \bar{u}$$

since $\alpha_\ell \to_T 0$ and $d^\ell \to \bar{d}$. Taking the limits in the previous inequality, we finally get

$$\gamma\langle \nabla_x q_{\tau_k}(\bar{u}, \bar{v}), \bar{d}\rangle \le \langle \nabla_x q_{\tau_k}(\bar{u}, \bar{v}), \bar{d}\rangle,$$

which is absurd since $\gamma \in (0, 1)$ and $\langle \nabla_x q_{\tau_k}(\bar{u}, \bar{v}), \bar{d}\rangle = -\nu < 0$. $\qquad\square$

In order to ensure that the sequence generated by the Alternating Minimization scheme properly converges, we need the sequence of directions $\{d_\ell\}$ to satisfy suitable properties. Here, in particular, we assume that the entire sequence of linear mappings $\{H_\ell\}$ satisfies the *bounded eigenvalues* condition [29, Sec. 1.2]:

$$c_1\|z\|^2 \le \langle z, H_\ell(z)\rangle \le c_2\|z\|^2 \qquad \forall z \in \mathbb{X}. \tag{4.4}$$

We are finally able to show that the inexact alternating minimization loop stops in a finite number of iterations providing a point (x^{k+1}, y^{k+1}) satisfying conditions (3.2)-(3.3).

Proposition 4.4 *Assume the sequence of linear maps $\{H_\ell\}$ in Algorithm 2 satisfies the bounded eigenvalues condition (4.4). Then the algorithm cannot cycle infinitely and determines in a finite number of iterations a point (x^{k+1}, y^{k+1}) such that*

$$\left\|\nabla_x q_{\tau_k}(x^{k+1}, y^{k+1})\right\| \le \delta_k$$

and

$$y^{k+1} = \operatorname*{argmin}_{y \in D} q_{\tau_k}(x^{k+1}, y).$$

Proof Suppose, by contradiction that, for some values of τ_k and δ_k, the sequence $\{(u^\ell, v^\ell)\}$ is infinite. From the instructions of the algorithm, it is possible to see that we have

$$q_{\tau_k}(u^{\ell+1}, v^{\ell+1}) \le q_{\tau_k}(u^0, v^0),$$

cf. (4.5). Hence, for all $\ell \geq 0$, the point (u^ℓ, v^ℓ) belongs to the level set

$$\{(u, v) \in \mathbb{X} \times \mathbb{X} \mid q_{\tau_k}(u, v) \leq q_{\tau_k}(u^0, v^0)\}.$$

Lemma 4.2 implies that this is a bounded set. Therefore, the sequence $\{(u^\ell, v^\ell)\}$ admits accumulations points. Let $K \subseteq \mathbb{N}$ be an infinite subset such that

$$\lim_{\substack{\ell \to \infty \\ \ell \in K}} (u^\ell, v^\ell) = (\bar{u}, \bar{v}).$$

Recalling the continuity of the gradient, we have

$$\lim_{\substack{\ell \to \infty \\ \ell \in K}} \nabla_x q_{\tau_k}(u^\ell, v^\ell) = \nabla_x q_{\tau_k}(\bar{u}, \bar{v}).$$

We now show that $\nabla_x q_{\tau_k}(\bar{u}, \bar{v}) = 0$. Taking into account the instructions of the algorithm, we have

$$q_{\tau_k}(u^{\ell+1}, v^{\ell+1}) \leq q_{\tau_k}(u^{\ell+1}, v^\ell) = q_{\tau_k}(u^\ell + \alpha_\ell d^\ell, v^\ell) < q_{\tau_k}(u^\ell, v^\ell). \qquad (4.5)$$

By (4.4), it is possible to see that

$$\|d^\ell\|^2 \leq c_2^2 \|\nabla_x q_{\tau_k}(u^\ell, v^\ell)\|^2.$$

Since $\nabla_x q_{\tau_k}(u^\ell, v^\ell) \to_K \nabla_x q_{\tau_k}(\bar{u}, \bar{v})$, we see that there exists a constant $M > 0$ such that $\|d^\ell\| \leq M$ for all $\ell \in K$.

Since the entire sequence $\{q_{\tau_k}(u^\ell, v^\ell)\}$ is monotonically decreasing by (4.5), and the subsequence $\{q_{\tau_k}(u, v)\}_K$ converges to $q_{\tau_k}(\bar{u}, \bar{v})$, it follows that the whole sequence of function values converges to this limit, i.e., we have

$$\lim_{\ell \to \infty} q_{\tau_k}(u^\ell, v^\ell) = q_{\tau_k}(\bar{u}, \bar{v}).$$

Hence (4.5) yields $\lim_{\ell \to \infty} q_{\tau_k}(u^\ell, v^\ell) - q_{\tau_k}(u^\ell + \alpha_\ell d^\ell, v^\ell) = 0$. Thus, the hypotheses of Lemma 4.3 are satisfied. Moreover, from (4.2) and (4.4), we have

$$\langle \nabla_x q_{\tau_k}(u^\ell, v^\ell), d^\ell \rangle \leq -c_1 \|\nabla_x q_{\tau_k}(u^\ell, v^\ell)\|^2.$$

Using Lemma 4.3, we therefore obain

$$0 = \lim_{\substack{\ell \to \infty \\ \ell \in K}} \langle \nabla_x q_{\tau_k}(u^\ell, v^\ell), d^\ell \rangle \leq \lim_{\substack{\ell \to \infty \\ \ell \in K}} -c_1 \|\nabla_x q_{\tau_k}(u^\ell, v^\ell)\|^2 \leq 0,$$

which implies that, for $\ell \in K$ sufficiently large, we have $\|\nabla_x q_{\tau_k}(u^\ell, v^\ell)\| \leq \delta_k$, i.e., that the stopping criterion of step (S.1) is satisfied in a finite number of iterations, and this contradicts the fact that $\{(u^\ell, v^\ell)\}$ is an infinite sequence. Condition (3.2) is then satisfied by the stopping criterion, whereas condition (3.3) follows by construction. $\qquad \square$

In order for the theoretical analysis to hold, we only need to ensure that H_ℓ satisfies condition (4.4). This assumption can be guaranteed a priori by different ways of defining H_ℓ. Among these valid choices, we can find classical setups leading back to iterations of standard algorithmic schemes such as gradient method ($H_\ell = I$), Newton method ($H_\ell = \nabla^2_{xx} q_{\tau_k}(u^\ell, v^\ell)$, provided f is uniformly convex), quasi-Newton methods and limited-memory BFGS type methods.

This aspect is crucial in practice: we are allowed to employ the most efficient solvers for nonlinear optimization to carry out step (S.3) of the Alternate Minimization algorithm and thus speed up the computation of step (S.2) of Algorithm 1, which is the most burdensome one. As a comparison, the Augmented Lagrangian algorithm from [1] has to resort to a gradient-based method to solve the (constrained) sequential subproblems, possibly resulting in an inefficient method especially for ill-conditioned problems. We will indeed observe this difference in Sect. 6. Another difference is pointed out in the following comment.

Remark 4.5 The Augmented Lagrangian algorithm from [1] has to compute projections onto the set D within the computation of the stepsizes, i.e., it may require many projections for a single (inner) iteration. This is a notable difference to our Algorithm 2, which requires only a single projection after the computation of the new iterate $u^{\ell+1}$. In fact, it would also be possible to apply several iterations of an unconstrained optimization solver to the subproblem of minimizing the penalty function $q_{\tau_k}(\cdot, v^\ell)$ before updating the v-component, i.e., before using a single projection step.

5 Particular instances

The idea of this section, similar to [1], is to present some difficult optimization problems where projections onto the complicated set D can be carried out easily. This section does not contain any proofs since the corresponding results are known from the literature. However, since these particular instances will be used in Sect. 6, they have to be discussed in some detail.

5.1 The case of sparsity constraints

A particular case of problem (1.1) is that of sparsity constrained optimization problems, i.e., optimization problems of the form

$$
\begin{aligned}
&\min_{x \in \mathbb{R}^n} \ f(x) \\
&\text{s.t. } G(x) \in C, \\
&\quad\quad x \in D = \{x \mid \|x\|_0 \le s\},
\end{aligned}
\tag{5.1}
$$

where $s < n$ and $\|x\|_0$ denotes the zero pseudo-norm of x, i.e., the number of nonzero components of x. The Penalty Decomposition approach was originally proposed in [14] for this class of problems, and the inexact version was then proposed for the case $\{x \mid G(x) \in C\} = \mathbb{R}^n$ [10].

In fact, from the analysis in Sect. 3, we can deduce that the convergence results continue to hold for the inexact version of the algorithm even in presence of additional constraints.

The Penalty Decomposition method is particularly appealing, from a computational perspective, for this class of problems since the Euclidean projection onto the sparse set D is easily obtainable in closed form, as outlined e.g. in [10, 14]. Let us denote the index set of the largest s variables at $\bar{x}$ in absolute value by $\mathcal{G}_s(\bar{x})$; for simplicity, we furthermore assume that cases of tie are handled unambiguously. Then, the projection of $\bar{x}$ onto D is given by

$$(\Pi_D(\bar{x}))_i = \begin{cases} \bar{x}_i & \text{if } i \in \mathcal{G}_s(\bar{x}), \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

In other words, the projection can be simply computed by setting to zero the $n - s$ smallest components of $\bar{x}$

Note that M-stationarity, as defined in Definition 2.1, coincides with Lu-Zhang stationarity [9], which is the property guaranteed to hold for cluster points obtained by the original Penalty Decomposition method [14]. Hence, we can conclude, from the results shown in Sect. 3, that the inexact Penalty Decomposition method has the same convergence properties as its exact counterpart, and that the M-stationarity concept includes a corresponding stationarity condition particularly designed for cardinality constrained problems. We note, however, that there exist further stationarity concepts in this setting, see the corresponding discussions in, e.g., [9, 14, 30, 31].

5.2 Low-rank approximation problems

Here we consider the space $\mathbb{X} = \mathbb{R}^{m \times n}$ with given $n, m \in \mathbb{N}$, $n, m \geq 2$; equipped with the standard Frobenius inner product, this is a Euclidean space.

In applications like computer vision, machine learning, computer algebra or signal processing, there is a strong interest in low-rank matrix optimization problems, see, e.g., [12, 32–35]. Specifically, letting $q = \min(m, n)$ and given $\kappa \leq q - 1$, we are interested in problems of the form

$$\begin{aligned} \min_{X \in \mathbb{R}^{m \times n}} \quad & f(X) \\ \text{s.t.} \quad & G(X) \in C, \\ & X \in D = \{X \mid \text{rank}(X) \leq \kappa\}. \end{aligned} \tag{5.3}$$

The set D has been thoroughly analyzed from a geometrical point of view, see e.g. [36] for a formula for $\mathcal{N}_D^{\lim}(X)$. Interestingly, elements of $\Pi_D(X)$ can be easily constructed exploiting the singular value decomposition of X [12, 15].

Proposition 5.1 *Let $X \in \mathbb{X} = \mathbb{R}^{m \times n}$ and let $X = U \Sigma V^T$ its singular value decomposition, with orthogonal matrices $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$ and $\Sigma \in \mathbb{R}^{m \times n}$ diagonal*

with entries in non-increasing order, i.e.,

$$\Sigma_{ij} = \begin{cases} \sigma_i & \text{if } i = j, \\ 0 & \text{otherwise}, \end{cases} \qquad \sigma_i \geq \sigma_j \; \forall i \geq j,$$

being $\sigma_1, \ldots, \sigma_q$ the singular values of X. Moreover, let $\hat{\Sigma}$ the matrix obtained setting to zero the $q - \kappa$ bottom-right elements of Σ, i.e.,

$$\hat{\Sigma}_{ij} = \begin{cases} \sigma_i & \text{if } i = j \leq \kappa, \\ 0 & \text{otherwise}. \end{cases}$$

Then, $\hat{X} = U\hat{\Sigma}V^T \in \Pi_D(X)$.

Of course, the computation of the SVD for a matrix X is not a costless operation, so obtaining an element of $\Pi_D(X)$, even though conceptually simple, requires a non-negligible amount of computing resources.

If we restrict the discussion to the case of symmetric positive semi-definite matrices, i.e., $D = \{X \in \mathbb{R}^{n \times n} \mid X \succeq 0, \; \text{rank}(X) \leq \kappa\}$, we can resort to the eigenvalue decomposition instead of the SVD [1, 15].

Proposition 5.2 *Let $X \in \mathbb{R}^{n \times n}$ be a symmetric matrix. Let us denote by $X = \sum_{i=1}^{n} \lambda_i v_i v_i^T$ its eigenvalue decomposition, where $\lambda_1 \geq \ldots \geq \lambda_n$ are the non-increasingly ordered eigenvalues with corresponding eigenvectors $v_1, \ldots, v_n$. Then, we have $\hat{X} = \sum_{i=1}^{\kappa} \max\{0, \lambda_i\} v_i v_i^T \in \Pi_D(X)$.*

We can thus observe that, in this particular case, in order to compute the projection onto the set D we only need to find the κ largest eigenvalues with the corresponding eigenvectors; this can be done efficiently, especially when κ is small, as in most applications.

A (exact) Penalty Decomposition scheme was developed in [15] to tackle low-rank optimization problems, exploiting the above closed form rules for projection onto D both in the general and the positive semi-definite cases. The analysis in Sect. 3 shows that the algorithmic framework maintains the same convergence properties even when the X-update step is carried out in an inexact fashion.

5.3 Box-switching constrained problems

A wide class of relevant optimization problems with difficult geometric constraints is constituted by the so called *box-switching* constrained problems [1] that can be formalized as follows:

$$\begin{aligned} \min_{x, y \in \mathbb{R}^n} \quad & f(x, y) \\ \text{s.t.} \quad & G(x, y) \in C, \\ & (x, y) \in D = \{(x, y) \mid x_i y_i = 0 \; \forall i, \; l_x \leq x \leq u_x, \; l_y \leq y \leq u_y\}, \end{aligned} \qquad (5.4)$$

where, for simplicity, we assume that $l_x \leq 0 \leq u_x$ and $l_y \leq 0 \leq u_y$.

This setting covers various disjunctive programming problems such as problems with

- switching constraints [8]: $l_x = l_y = -\infty$, $u_x = u_y = \infty$,
- complementarity constraints [6]: $l_x = l_y = 0$, $u_x = u_y = \infty$,
- relaxed sparsity constraints [37]: $l_x = -\infty$, $u_x = \infty$, $l_y = 0$, $l_y = 1$.

It is easy to realize that projection onto the set D in this case is simple. Indeed, let us first consider the projection onto classical bound constraints $[l, u]$ of a vector w. Since the constraints are separable, we can immediately obtain the projection by computing, for each component i, the value

$$
(P_{[l,u]}(w))_i = \begin{cases} w_i & \text{if } l_i \leq w_i \leq u_i, \\ l_i & \text{if } w_i < l_i, \\ u_i & \text{if } w_i > u_i. \end{cases}
$$

With this in mind, noting that the set D is also (pairwise) separable, we can obtain an element $(\hat{x}, \hat{y}) \in \Pi_D[(\bar{x}, \bar{y})]$ by first computing

$$
\tilde{x} = P_{[l_x, u_x]}(\bar{x}), \qquad \tilde{y} = P_{[l_y, u_y]}(\bar{y})
$$

and then setting

$$
(\hat{x}_i, \hat{y}_i) = \begin{cases} (\tilde{x}_i, 0) & \text{if } \bar{x}_i^2 + (\tilde{y}_i - \bar{y}_i)^2 \geq (\tilde{x}_i - \bar{x}_i)^2 + \bar{y}_i^2, \\ (0, \tilde{y}_i) & \text{otherwise.} \end{cases}
$$

Computing the projection onto D thus amounts to computing $2n$ projections onto real intervals, which can be done with low computational effort. For this reason, a Penalty Decomposition type scheme again appears particularly appealing for this class of problems.

5.4 General disjunctive programs

A broad class of optimization problems with geometric constraints is represented by programs where variables are required to satisfy at least one among several sets of constraints:

$$
\begin{aligned}
\min_{x \in \mathbb{R}^n} \quad & f(x) \\
\text{s.t.} \quad & G(x) \in C, \\
& x \in D = \bigcup_{i=1}^N D_i,
\end{aligned} \tag{5.5}
$$

where D_i, $i = 1, \ldots, N$ are closed convex sets. The resulting overall feasible set of these *disjunctive programming problems* [4] typically takes the structure of a nonconvex, disconnected set. Projections onto D in this case can be computed by finding the

closest among the N projections onto $D_1, \ldots, D_N$:

$$\Pi_D(x) = \underset{z}{\operatorname{argmin}}\{\|z - x\| \mid z = P_{D_i}(x), \; i = 1, \ldots, N\}.$$

Since, in general, the projection onto a convex set is already an expensive operation, the projection onto D is consequently a costly task. We shall observe that, in fact, the settings analyzed in the previous subsections are particular instances of this setting where the peculiar structure of sets D_i allows to efficiently compute the projection in smart ways.

The Penalty Decomposition approach might be appealing for problems of this form when the constraints $G(x) \in C$ are numerous and/or nontrivial and N is also large. In these cases, the brute force strategy of solving N problems with convex constraints may become computationally unsustainable and PD might represent an appealing alternative.

6 Computational experiments

In this section, we report the results of an extensive experimentation aimed at demonstrating the potential and the benefits of using the Penalty Decomposition algorithm on various classes of problems. The experiments have two main goals:

- Analyze the behavior of penalty decomposition in different settings and understand how to make it as efficient as possible;
- Compare the penalty decomposition approach with the augmented Lagrangian method proposed in [1], which is, to the best of our knowledge, the only available algorithm from the literature designed to handle the general setting (1.1).

The code for the experiments has entirely been implemented in Python 3.9 and is publicly available at https://github.com/MatteoLapucci/GeoIPD. All the experiments have been run on a machine with the following specifications: Intel Xeon Processor E5-2430 v2, 6 physical cores (12 threads), 2.50 GHz, 16 GB RAM. We considered benchmarks of problems from the classes discussed in Sect. 5, i.e., cardinality constrained problems, low-rank approximation problems and disjunctive programming problems.

For the Penalty Decomposition approach (Algorithm 1), we set an upper bound to the value of τ_k equal to 10^8. We employed for the inner loop (Algorithm 2) the stopping criterion

$$q_{\tau_k}(u^\ell, v^\ell) - q_{\tau_k}(u^{\ell+1}, v^{\ell+1}) \leq \epsilon_{\text{in}}, \tag{6.1}$$

whereas for the outer loop we employ

$$\|x^{k+1} - y^{k+1}\| + \operatorname{dist}_C(G(x^{k+1})) \leq \epsilon_{\text{out}}.$$

Both the above stopping conditions are the ones suggested in [14] and we set $\epsilon_{\text{in}} = 10^{-5}$ and $\epsilon_{\text{out}} = 10^{-5}$.

As unconstrained optimization solvers for the x-update step, we implemented the gradient descent algorithm with Armijo line search. We also ran experiments using the implementations of the conjugate gradient (CG, [29]), BFGS [29] and L-BFGS [38] methods available in the `scipy` library. For all of these algorithms, we used the stopping criterion $\|\nabla_x q_{\tau_k}(u^{\ell+1}, v^\ell)\| \le \epsilon_{\mathrm{solv}}$, with $\epsilon_{\mathrm{solv}} = 10^{-5}$ if not specified otherwise.

As for the augmented Lagrangian method (ALM) from [1], it employs as inner solver of subproblems the spectral gradient method (SGM) proposed in the same work. With reference to [1, Algorithm 3.1], we set $\sigma = 10^{-5}$, $\gamma_0 = 1$, $\gamma_{\max} = 10^{12}$, $m = 10$, $\tau = 2$ (note that here it does not denote the penalty parameter). As for the ALM ([1, Algorithm 4.1]), we set $\eta = 0.8$. We employed the multipliers safeguarding technique, projecting the values obtained using the standard Hestenes-Powell-Rockafellar updates onto the box $[-10^8, 10^8]$. For the spectral gradient loop, we used the stopping condition

$$\max_{j=0,\ldots,m-1} q_{\tau_k}(x^{\ell-m}) - \min_{j=0,\ldots,m} q_{\tau_k}(x^{\ell+1-m}) \le 10^{-5},$$

where here we have used the notation of the present paper. We used the same stopping condition (6.1) as the PD method with $\epsilon_{\mathrm{in}} = 10^{-5}$ for the inner loop of the ALM, whereas for the outer loop we require $\mathrm{dist}_C(G(x^{k+1})) \le \epsilon_{\mathrm{out}}$, with $\epsilon_{\mathrm{out}} = 10^{-5}$. The stopping conditions have been chosen as similar as possible for the two algorithms, in order to have a fair comparison.

We also did experiments with a variant of our proposed approach, employing safe-guarded Lagrange multipliers in an augmented Lagrangian fashion, i.e., we take the ALM approach from [1] and combine it with the decomposition idea to solve the resulting subproblems. In other terms, the penalty function in this case becomes:

$$q_\tau(x, y; \lambda_G, \lambda_E) = f(x) + \frac{\tau}{2}\mathrm{dist}_C^2\left(G(x) + \frac{\lambda_G}{\tau}\right) + \lambda_E^T(x - y) + \frac{\tau}{2}\|x - y\|^2.$$

The setting of multipliers and penalty parameter updates is the same as the one we employed for the ALM itself. In the following, we will show that this modification (denoted PDLM), which does not have any major impact in the convergence analysis, leads to significant benefits in practice. It is interesting to note that this finding is in contrast with the remarks that can be found in the conclusions of [14].

Finally, we point out that we did not report the values for the initial penalty parameter τ_0 and its growing rate α_τ (we always set $\tau_{k+1} = \alpha_\tau \tau_k$). Indeed, these parameter are quite crucial for the overall performance of both PD and ALM algorithms and have been suitably selected for each class of problems. Thus, we will report each time the specific values of these two parameters.

6.1 Sparsity constrained optimization problems

We begin our numerical analysis with sparsity constrained problems. The reason we start with this class of problems is twofold: a) the original PD approach was

Table 1 Results of experiments on three random instances of cardinality constrained quadratic problems (6.2). Problems were generated according to (6.3), (6.4) with $n_{cond} = 10$, $s = 3$, $v = 5$

n	Algorithm	f_val	Runtime (s)
10	PD_gd	−9.70	3.68
	PD_bfgs-scipy	−9.70	0.53
	PD_lbsfgs-scipy	−9.70	0.31
	PDLM_lbsfgs-scipy	−9.63	0.26
	SGM	−9.63	0.05
25	PD_gd	−15.12	5.96
	PD_bfgs-scipy	−15.12	0.91
	PD_lbsfgs-scipy	−15.12	0.54
	PDLM_lbsfgs-scipy	−16.19	0.24
	SGM	−15.58	0.01
50	PD_gd	−23.92	9.17
	PD_bfgs-scipy	−23.92	1.69
	PD_lbsfgs-scipy	−23.92	0.89
	PDLM_lbsfgs-scipy	−23.92	0.31
	SGM	−23.92	0.01

designed for these problems and b) results are more easily and intuitively interpretable.

We considered various experimental settings with problems of this class. Firstly, we begin with the simplest possible problems, i.e., convex quadratic problems with only sparsity constraints:

$$\min_x \frac{1}{2} x^T Q x + v c^T x \quad \text{s.t. } \|x\|_0 \leq s. \tag{6.2}$$

We randomly generated instances of problem (6.2), according to the following procedure:

$$Q = YDY, \quad Y = I - \frac{2}{\|y\|^2} yy^T, \quad y \in \mathbb{R}^n : y_i \sim \mathcal{U}(-1, 1), \tag{6.3}$$

$$D = \text{diag}(d_1, \ldots, d_n), \quad d_i = \exp\left(\frac{i-1}{n-1} n_{cond}\right), \quad c \in \mathbb{R}^n : c \sim \mathcal{U}(-1, 1) \tag{6.4}$$

where n_{cond} denotes the desired condition number of the matrix Q. We generated three instances with $n_{cond} = 10$, $n \in \{10, 25, 50\}$, $s = 3$, $v = 5$ to evaluate the impact of different solvers for the x-update step on the alternating minimization scheme and, in turn, on the overall PD approach.

We report in Table 1 the results obtained by running PD equipped with different inner solvers starting from the origin. We also ran the variant with Lagrange multipliers of our approach only with L-BFGS as inner solver; here we set $\tau_0 = 1$ and $\alpha_\tau = 1.1$. Moreover, we considered the spectral gradient method for comparison. Note that, since there are no additional constraints, there is no need to resort to the ALM.

Table 2 Convergence of penalty decomposition methods on [30, Example 2.2] for different values of τ_0. We report the number of times an objective value has been obtained out of 1000 runs from different starting points chosen randomly in $[-10, 10]^n$

τ_0	Algorithm	$f = -41.33$	$f = -39$	$f = -36.33$	Others
0.1	PD	1000	0	0	0
	PDLM	1000	0	0	0
1	PD	661	339	0	0
	PDLM	1000	0	0	0
10	PD	683	275	42	0
	PDLM	620	326	54	0
100	PD	549	250	40	161
	PDLM	527	295	57	121

As expected, the use of quasi-Newton type solvers is highly beneficial: BFGS leads to much faster convergence than the simple gradient method; the L-BFGS provides an additional, substantial speed up. The presence of Lagrange multipliers also seems to be beneficial, both in terms of efficiency and of quality of the obtained solution. Based on these result, in the following we will always be using L-BFGS for the x-update step in Algorithm 2.

Note that the spectral gradient method clearly outperforms the PD approach in this case. This is indeed not surprising: being there no additional constraint, there is no need with the SGM to adopt a sequential penalty strategy, which is costly.

Next, we turn to a simple verification of the convergence properties of the PD approach. In particular, we consider the artificial example [30, Example 2.2], which is an instance of (6.2) with $Q = E + I$, being E the matrix of all ones, $c = -(3, 2, 3, 12, 5)^T$, $v = 1$ and $s = 2$. We ran both PD and PDLM, with $\alpha_\tau = 1.1$ from 1000 different starting points randomly generated in the hyper-box $[-10, 10]^5$. We observed that the result strongly depends on the choice of τ_0, as we report in Table 2. Interestingly, both algorithms always converged to the global minimum $f(x^\star) = -41.33$ when we set $\tau_0 = 0.1$; in fact, we observed the same result for smaller values of τ_0. On the other hand, as τ_0 grows worse local minimizers become increasingly probable; the presence of Lagrange multipliers seems to alleviate, but not to suppress, this inconvenience. We argue that large values of τ_0 make PD schemes more dependent on the starting point: since usually $x^0 = y^0$, the penalty term is at the first iteration equal to $\tau_0 \|x - x^0\|^2$, which binds variable x close to the start.

At this point, we have devised a setting that apparently makes the PD approach efficient and effective. We therefore expect the algorithm to indeed be a good choice to resort to when: a) additional constraints are present and/or b) the projection operator is costly. In the former case, SGM needs to be employed within another sequential scheme, namely, the ALM, which is the only alternative to the PD available from the literature; in the latter case, the advantage of PD over the ALM may not be straightforward. In fact, the two algorithms share a similar structure, sequentially solving penalized subproblems; in order to do so, unconstrained continuous optimization steps and projections onto D are repeatedly carried out; however, in PD many descent steps

can be carried out before turning to the projection step; on the contrary, in the ALM we need to do the projection after every gradient step (in fact, we do it many times per iteration of the SGM to satisfy the acceptance criterion), cf. the discussion in Remark 4.5.

We now turn to sparsity constrained problems with additional constraints. In particular, we keep considering convex quadratic problems, but with simplex constraints, i.e., problems of the form

$$\min_x \ \frac{1}{2}x^T Q x + v c^T x, \quad \text{s.t.} \ e^T x = 1, \quad x \geq 0, \quad \|x\|_0 \leq s,$$

where $e \in \mathbb{R}^n$ denotes the vector of all ones. This is a classical sparse portfolio optimization problem [39], where Q and c denote the covariance matrix and the mean of n possible assets.

Portfolio optimization problems are particularly useful to test the proposed algorithm since we can easily obtain the global optimum to be used as a reference. Indeed, we can do so exploiting the mixed-integer reformulation of the problem with binary indicator variables $z_i \in \{0, 1\}$, $i = 1, \ldots, n$, and linear constraints $0 \leq x_i \leq z_i$, $e^T z \leq s$, and employing efficient software solvers such as Gurobi [40].

We first consider synthetic problems. Using (6.3), (6.4), we generated 10 problems for each combination of $n \in \{20, 40, 60\}$ and $n_{\text{cond}} \in \{10, 100, 500\}$, for a total of 90 problems. We set $s = 4$ when $n = 20$, $s = 7$ for $n = 40$ and $s = 9$ for $n = 60$. We set $v = 1$ and use as starting point of the experiments $\tilde{x} = (1/n, \ldots, 1/n)^T$; we ran PD, PDLM, ALM all with $\tau_0 = 1$ and $\alpha_\tau = 1.1$. We also ran Gurobi on all instances to obtain the global optimizer to be used as reference; note that Gurobi indeed finds the certified global minimum in tens of seconds. The overall results of the experiments are reported in Fig. 1. The results concerning efficiency (runtime) are presented in the form of performance profiles [41] in Fig. 1a. We can observe that Penalty Decomposition with Lagrange multipliers is generally faster than the other two considered algorithms. As for the quality of the retrieved solutions, we plot in Fig. 1b the cumulative distribution of the relative gap between the solution found by a solver and the certified global optimum found with Gurobi; the result of PDLM is surprisingly remarkable, as it almost always reached a value very close, and often equal to, the global optimum; on the contrary, both PD and the ALM end up with substantially suboptimal solutions in almost a half of the cases.

We conclude the analysis on sparsity constrained problems looking at the results on 6 instances of real world portfolio selection problems. In particular, the data used in the experiments consists of daily data for securities from the FTSE 100 index, from 01/2003 to 12/2007. The three datasets are referred to as DTS1, DTS2, and DTS3, and are formed by $n = 12, 24$, and 48 securities, respectively. We also included three datasets from the Fama/French benchmark collection (FF10, FF17, and FF48, with n equal to 10, 17, and 48), using the monthly returns from 07/1971 to 06/2011. The datasets are generated as in [42]. For each dataset, we define an instance of problem (6.1): the values of s and v are set as reported in Table 3, and are such that the cardinality constraint is active at the optimal solution. We used again $\tilde{x} = (1/n, \ldots, 1/n)^T$ as starting point. As for the penalty parameter, we set $\tau_0 = 0.01$ and for the Penalty
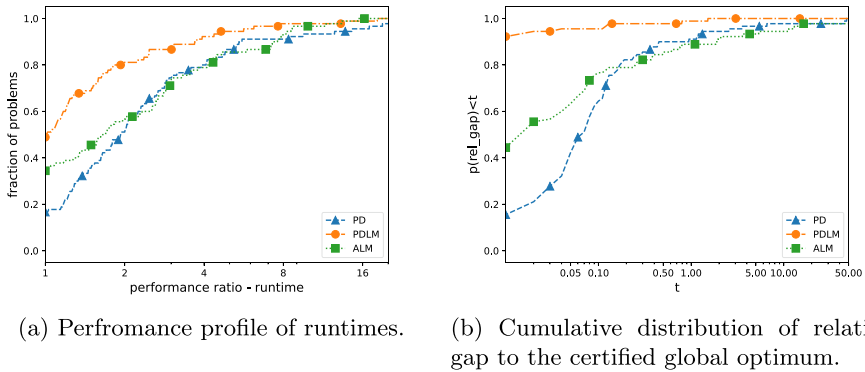
(a) Perfromance profile of runtimes.

(b) Cumulative distribution of relative gap to the certified global optimum.

Fig. 1 Results of experiments on 90 randomly generated sparse portfolio selection problems, running PD, PDLM and the ALM

Decomposition methods, whereas we found that a larger value $\tau_0 = 1$ was beneficial for the ALM. The parameter α_τ was set to 1.01 for all methods. The results are reported in Table 3 and we can observe that the trends outlined by the previous experiments are substantially confirmed.

6.2 Low-rank optimization problems

In this section, we study problems as discussed in Sect. 5.2 where $\mathbb{X} = \mathbb{R}^{m \times n}$ and D consists of a low-rank matrices space.

To begin with, we consider the class of *nearest low-rank correlation matrix problems*, which was already used as a benchmark in [15]. In detail, the problem can be formulated as

$$\min_{X \in \mathbb{R}^{n \times n}} \frac{1}{2} \|X - A\|_F^2 \quad \text{s.t.} \quad X^T = X, \ X \succeq 0, \ \text{diag}(X) = e, \ \text{rank}(X) \leq \kappa,$$

where A is a given symmetric correlation matrix. The test problems we consider are the same as in [15], and their corresponding matrix A is defined as follows:

- (P1) $A_{ij} = 0.5 + 0.5 \exp(-0.05|i - j|)$ for all i, j;
- (P2) $A_{ij} = \exp(-|i - j|)$ for all i, j;
- (P3) $A_{ij} = 0.6 + 0.4 \exp(-0.1|i - j|)$ for all i, j.

For each of the above problems, we considered the instances with $n = 200$ and $n = 500$ and a value of $\kappa = 5, 10$ and 20.

We experimentally compared the ALM and some implementations of the Penalty Decomposition approach; for all these algorithms, we set $\tau_0 = 1$ and $\alpha_\tau = 1.2$. We also needed for these experiments to set the upper bound on the value of τ_k to 10^{12}.

Note that solving the X-update subproblem with an iterative solver has a significant cost, as we are considering problems with up to $n \times n = 250000$ variables; moreover, we are dealing with ill-conditioned quadratic problems, thus we found convenient switching from L-BFGS to the CG method. We tested two settings for the

Table 3 Results of experiments on 6 real world sparse portfolio selection problems, solved using different algorithmic approaches

Problem (s, ν)	Algorithm	f_val	Runtime (s)
DTS1 (2, 0.001)	Gurobi	4.10e−05	0.02
	PD	4.27e−05	2.01
	PDLM	4.25e−05	2.07
	ALM	4.69e−05	8.72
DTS2 (4, 0.001)	Gurobi	2.52e−05	0.04
	PD	2.75e−05	2.49
	PDLM	2.75e−05	1.55
	ALM	2.75e−05	5.36
DTS3 (6, 0.001)	Gurobi	2.19e−05	0.19
	PD	2.43e−05	4.30
	PDLM	2.42e−05	1.72
	ALM	2.48e−05	5.98
FF10 (2, 0.05)	Gurobi	2.87e−05	0.01
	PD	3.07e−05	2.43
	PDLM	2.96e−05	3.38
	ALM	2.87e−05	7.21
FF17 (2, 0.05)	Gurobi	2.08e−05	0.01
	PD	3.34e−05	3.26
	PDLM	3.11e−05	3.98
	ALM	3.46e−05	47.89
FF48 (5, 0.05)	Gurobi	−1.10e−05	0.06
	PD	1.59e−05	11.09
	PDLM	−9.30e−06	5.49
	ALM	9.66e−05	0.47

X-update step with CG: a strongly inexact setting, where the CG method is stopped after at most 5 steps (`PD-cg-inaccurate`), or when the norm of the gradient is smaller than $\epsilon_{\text{solv}} = 0.1$, and a more accurate setting, where up to 20 CG steps are carried out and the tolerance for the gradient norm stopping condition is set to 0.001 (`PD-cg-accurate`).

In addition, we note that, in fact, the X-update subproblem

$$\min_{\substack{X \in \mathbb{R}^{n \times n} \\ X = X^T}} \frac{1}{2} \left\| X - A \right\|_F^2 + \frac{\tau}{2} \left(\left\| X - Y \right\|_F^2 + \left\| \text{diag}(X) - e \right\|^2 \right)$$

can be solved to global optimality in closed form; we thus also carried out experiments with this option (`PD-exact`); moreover, we also consider the strategy adopted in [15], where the constraint $\text{diag}(X) = e$ is kept as a lower-level constraint and the X-update subproblem is still solved in closed form (`PD-exact-lower-level`).

Table 4 Results of experiments on nearest low-rank correlation matrix problem (P1); for each instance we report the runtime, objective value and number of (inner) iterations for each considered solver

Problem (n, κ)	Algorithm	f_val	Runtime (s)	iter
P1(200, 5)	PD-cg-inaccurate	183.8	19.6	1539
	PD-cg-accurate	183.7	104.3	8470
	PD-exact	183.7	37.8	9179
	PD-exact-lower-level	183.7	20.2	6494
	ALM	183.7	124.8	23583
P1(200, 10)	PD-cg-inaccurate	27.7	9.5	597
	PD-cg-accurate	27.6	22.11	3277
	PD-exact	27.6	22.7	3587
	PD-exact-lower-level	27.6	12.4	2606
	ALM	27.6	25.8	3705
P1(200, 20)	PD-cg-inaccurate	3.7	8.7	273
	PD-cg-accurate	3.5	37.1	1194
	PD-exact	3.5	13.7	1332
	PD-exact-lower-level	3.5	7.5	1006
	ALM	3.5	31.1	2261
P1(500, 5)	PD-cg-inaccurate	3108.0	466.7	5602
	PD-cg-accurate	3107.0	2396.3	29689
	PD-exact	3107.0	619.4	33654
	PD-exact-lower-level	3107.0	339.4	23053
	ALM	3107.0	1710.1	47539
P1(500, 10)	PD-cg-inaccurate	748.2	543.5	2132
	PD-cg-accurate	748.2	3410.9	12673
	PD-exact	748.2	356.2	14299
	PD-exact-lower-level	748.2	207.6	9846
	ALM	748.2	1351.5	17322
P1(500, 20)	PD-cg-inaccurate	123.7	200.2	811
	PD-cg-accurate	123.4	1425.6	5078
	PD-exact	123.4	216.0	5787
	PD-exact-lower-level	123.4	127.5	4077
	ALM	123.4	1201.9	13568

We finally report that we found the introduction of Lagrange multipliers associated with constraints $G(x) \in C$ useful. We instead noticed that multipliers associated with the constraint $X = Y$ are not helpful. This observation is in line with the work in [15], where only the constraint $X = Y$ was in practice handled by the penalty approach and multipliers were reported not to be beneficial. In the experiments described in the following, only multipliers associated with the original problem constraints have been employed. The results of the experiment are reported in Tables 4, 5 and 6.

We can observe that the exact versions of the PD approach are the best performing ones from all perspectives, with the PD-exact-lower-level originally used in

Table 5 Results of experiments on nearest low-rank correlation matrix problem (P2); for each instance we report the runtime, objective value and number of (inner) iterations for each considered solver

Problem (n, κ)	Algorithm	f_val	Runtime (s)	iter
P2(200, 5)	PD-cg-inaccurate	3701.2	51.3	4312
	PD-cg-accurate	3700.8	181.7	15415
	PD-exact	3700.8	63.4	16360
	PD-exact-lower-level	3700.8	32.6	10724
	ALM	3700.8	156.4	22519
P2(200, 10)	PD-cg-inaccurate	1703.7	29.8	2162
	PD-cg-accurate	1703.1	100.85	7678
	PD-exact	1703.1	45.0	8171
	PD-exact-lower-level	1703.1	24.1	5384
	ALM	1703.1	136.7	14017
P2(200, 20)	PD-cg-inaccurate	712.2	30.9	1065
	PD-cg-accurate	712.0	118.2	3738
	PD-exact	712.0	35.0	3969
	PD-exact-lower-level	712.0	19.2	2629
	ALM	712.0	117.1	6519
P2(500, 5)	PD-cg-inaccurate	24249.5	950.8	11449
	PD-cg-accurate	24248.2	3463.7	42892
	PD-exact	24248.2	836.4	46711
	PD-exact-lower-level	24248.2	447.9	30240
	ALM	24248.2	627.6	15633
P2(500, 10)	PD-cg-inaccurate	11752.9	501.5	5754
	PD-cg-accurate	11749.1	1845	21508
	PD-exact	11749.1	560.9	23576
	PD-exact-lower-level	11749.1	308.4	15317
	ALM	11749.1	2289.7	44857
P2(500, 20)	PD-cg-inaccurate	5505.0	283.1	2878
	PD-cg-accurate	5502.9	1049.6	12854
	PD-exact	5502.9	420.1	11932
	PD-exact-lower-level	5502.9	238.6	7834
	ALM	5502.9	2642.8	360565

[15] standing out. This is in fact not surprising: in this case the exact method solves subproblems not only with higher accuracy, but also employing much less time than using an iterative solver.

Interestingly, however, we observe that the "inaccurate" version of the inexact PD attains runtimes that are comparable with the exact approaches, with only small drops in the quality of the retrieved solution. On the other hand, with a slightly more accurate inexact minimization we are always able to retrieve the best solution as the exact methods, with a computational effort generally comparable to that of the ALM.

Table 6 Results of experiments on nearest low-rank correlation matrix problem (P3); for each instance we report the runtime, objective value and number of (inner) iterations for each considered solver

Problem (n, κ)	Algorithm	f_val	Runtime (s)	iter
P3(200, 5)	PD-cg-inaccurate	265.1	22.2	1746
	PD-cg-accurate	265.0	108.7	9224
	PD-exact	265.0	40.3	9934
	PD-exact-lower-level	265.0	21.5	6937
	ALM	265.0	131.6	21078
P3(200, 10)	PD-cg-inaccurate	56.1	24.2	704
	PD-cg-accurate	56.1	103.7	3277
	PD-exact	56.1	24.8	3587
	PD-exact-lower-level	56.1	13.5	2606
	ALM	56.1	82.6	3705
P3(200, 20)	PD-cg-inaccurate	9.1	9.29	305
	PD-cg-accurate	8.5	45.4	1474
	PD-exact	8.5	16.1	1625
	PD-exact-lower-level	8.5	8.9	1196
	ALM	8.5	109.9	7875
P3(500, 5)	PD-cg-inaccurate	2871.4	1451.3	5607
	PD-cg-accurate	2869.3	7902.8	29897
	PD-exact	2869.3	615.6	34110
	PD-exact-lower-level	2869.3	360.8	23229
	ALM	2869.3	300.57	4128
P3(500, 10)	PD-cg-inaccurate	982.1	219.9	2460
	PD-cg-accurate	981.8	1182.7	13657
	PD-exact	981.8	371.1	16322
	PD-exact-lower-level	981.8	209.9	10463
	ALM	981.8	444.7	9343
P3(500, 20)	PD-cg-inaccurate	243.8	102.2	964
	PD-cg-accurate	243.7	1049.6	10757
	PD-exact	243.7	420.1	11932
	PD-exact-lower-level	243.7	238.6	7834
	ALM	243.7	2642.8	36056

We can thus deduce that a suitable configuration exists for the inexact PD approach that provides a good trade-off between solution quality and runtime.

These results are encouraging for all those settings where the exact version of the Penalty Decomposition approach is not employable by construction.

We then turn to a new class of problems, where matrices are not symmetric positive semi-definite and the X-update step requires a solver to be carried out. Specifically, we consider the *low-rank based multi-task training* [43] *of logistic models* [44]. Given a collection of somewhat related binary classification tasks $\mathcal{T}_1, \ldots, \mathcal{T}_m, \mathcal{T}_i = \{(X_i, Y_i) \mid$

$X_i \in \mathbb{R}^{N_i \times n}$, $Y_i \in \{0, 1\}^{N_i}\}$, where X_i represents the data matrix for each task and Y_i the corresponding labels, we can formalize the multitask logistic regression training problem as

$$\min_{W, U, V \in \mathbb{R}^{m \times n}} \sum_{t=1}^{m} \mathcal{L}(W_t; X_t, Y_t) + \eta \|U\|_F^2 \quad \text{s.t. } \text{rank}(V) \leq \kappa, \quad W = U + V, \quad (6.5)$$

where the t-th row W_t of W denotes the weights of the logistic model for the t-th task; each model is defined as the sum of a component independently characterizing the particular task, which is regularized, and a second component that lies in a linear subspace shared by all tasks. The stronger is the regularization parameter η, the higher will be the similarity of the obtained models. By $\mathcal{L}(W_t; X_t, Y_t)$ we denote the binary cross entropy loss function of the logistic model for task t, which is a convex function that, however, cannot be minimized in closed form.

We can observe that the problem can be solved by Penalty Decomposition, duplicating variable V. The constraint $W = U + V$ can also be tackled by the penalty approach. The update step of the original variables W, U, V cannot be carried out in closed form, thus we need to resort to the inexact version of the PD method.

For the experiments, we used the `landmine` dataset [45], consisting of 9-dimensional data points representing radar images from 29 landmine fields/tasks. Each task aims to classify points as landmine or clutter. There are 14,820 data points in total. Tasks can approximately be clustered into two classes of ground surface conditions, so we expect $r = 2$ to be a reasonable bound for the low-rank component of the solution. We defined four instances of problem (6.5), corresponding to values of η in $\{0.01, 0.1, 0.5, 2\}$. We examined the behavior of the inexact PD and ALM methods under different parameters configurations. In particular, we considered the following settings:

- Penalty decomposition

    - Lagrange multipliers associated with all constraints;
    - $\tau_0 = 10^{-3}$, $\alpha_\tau = 1.3$;
    - conjugate gradient (CG) for x-update steps;
    - three options for CG termination criteria:
        - $\epsilon_{\text{solv}} = 0.1$, max_iters$_{\text{CG}} = 5$ (`pd_inaccurate`);
        - $\epsilon_{\text{solv}} = 0.05$, max_iters$_{\text{CG}} = 8$ (`pd_mid`);
        - $\epsilon_{\text{solv}} = 0.001$, max_iters$_{\text{CG}} = 20$ (`pd_accurate`);

- ALM

    - $\tau_0 = 1$, $\alpha_\tau = 1.3$;
    - two options for spectral gradient parameters:
        - $\epsilon_{\text{in}} = 10^{-1}$, $m = 1$, $\gamma_{\text{max}} = 10^6$, $\sigma = 0.05$ (`alm_fast`);
        - $\epsilon_{\text{in}} = 10^{-3}$, $m = 4$, $\gamma_{\text{max}} = 10^9$, $\sigma = 5 \cdot 10^{-4}$ (`alm_accurate`).

We also report the results obtained by optimizing each task independently. For all PD and ALM configurations, we used as starting solution the one retrieved by single task optimization. Note that both configurations for ALM have lower precision than the
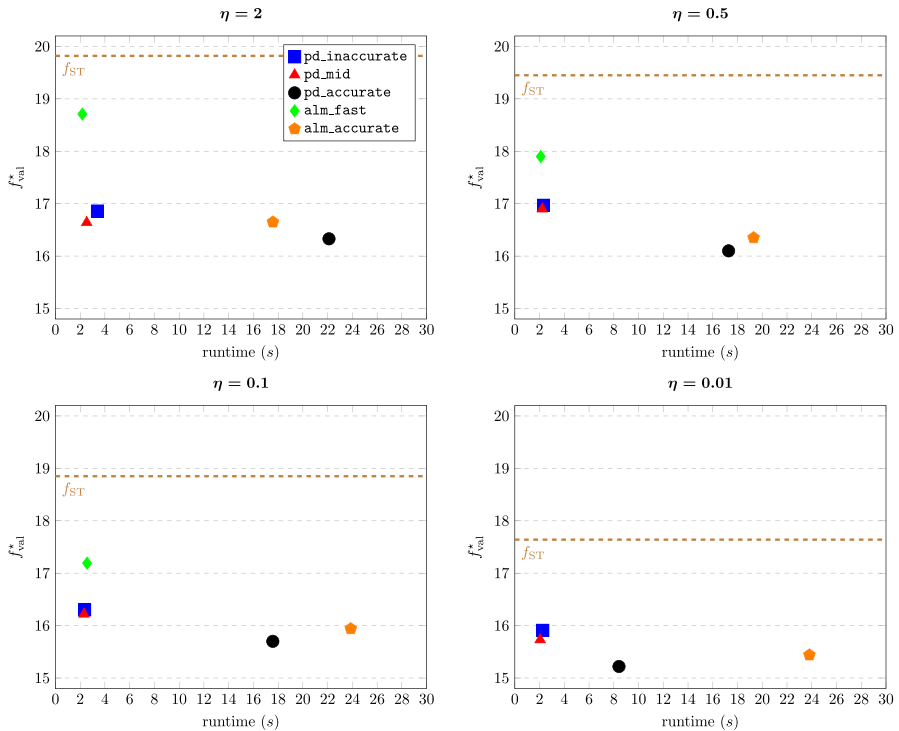
Fig. 2 Runtime/quality trade-off for different set-ups of PD and ALM on low-rank multitask logistic regression problems. The four problems are obtained from the `landmine` dataset for different values of the regularization parameter η and setting $\kappa = 2$

default one reported at the beginning of Sect. 6; with this particular problem, we found the default configuration to be remarkably inefficient; however we shall underline that, in other test cases considered in this paper, these alternative configurations had led to convergence issues concerning numerical errors. The results of the experiment are reported in Fig. 2. Note that here we are interested in the optimization process metrics, not in the out-of-sample prediction performance of the obtained models.

We can observe that different setups for the algorithms allow to obtain different trade-offs between speed and solution quality. In particular, for the PD method we see that the trend observed with the correlation matrix problems are confirmed: solving the x-update subproblem up to lower accuracy allows to save computing time but at the cost of small yet not negligible sacrifice on the solution quality. A similar and even stronger trend can be observed for the ALM. Finally, we can observe that PD appears to be superior to the ALM both in terms of efficiency and effectiveness.

6.3 Disjunctive programming problems

In this section, we computationally analyze the performance of the Penalty Decomposition approach on problems of the form (5.5). The main goal of this section is to

compare the performance of inexact PD with that of the ALM algorithm in a setting where the projection operation is costly and is in fact responsible for the largest part of the computational burden: as highlighted in Sect. 5.4, it amounts to compute the projection onto each of the convex sets D_i.

For the experiments, we defined the following test problem:

$$\min_{x \in \mathbb{R}^n} \mathcal{L}(x)$$

$$\text{s.t. } x \in \bigcup_{q=1}^{N} \{x \mid A_q x \le b_q\}$$

$$\sum_{i=1}^{n} c_{ij}(x_i - p_{ij})^4 \le t_j, \quad j = 1, \ldots, m,$$

where $\mathcal{L}$ denotes the (convex) logistic loss on a randomly generated dataset of 200 examples. We assume $A_q \in \mathbb{R}^{s \times n}$ and $b_q \in \mathbb{R}^s$ have the same dimensions for $q = 1, \ldots, N$ and their coefficients are uniformly drawn from $[-1, 1]$; the coefficients c are randomly picked from $[0, 1]$, whereas values for p are from $[-0.5, 0.5]$. We set $t_j = 0.1$ for all j.

First, we consider the problem with $n = 10$, $s = 12$ and $m = 1$, for values of N varying in $\{2, 5, 10, 20, 50, 100\}$. In Table 7 we report the results obtained running PD (parameters: $\tau_0 = 0.1$, $\alpha_\tau = 1.2$, $\epsilon_{\text{in}} = 0.01$, Lagrange multipliers employed) and the ALM (parameters $\tau_0 = 1$, $\alpha_\tau = 1.2$ $m = 4$, $\sigma = 0.01$, $\epsilon_{\text{in}} = 0.01$), together with reference values obtained with the enumeration approach (subproblems are solved using the SLSQP method available in `scipy`), which allows to retrieve the certified global optimizer. For the projection steps onto sets D_i we used `gurobi` solver.

We can observe that PD was always much faster than the ALM; moreover, it always ended up finding the actual global optimizer; this does not hold true for the ALM. We remark that we verified that, as expected, the computing time is indeed entirely dominated by projection steps. Indeed, as can be observed from Table 7, runtime and number of projection steps appear to be somewhat correlated.

Then, we turn to the experiments on an instance where the number of nonlinear constraints shared by all components of the feasible set are numerous and dominate the complexity of solving each subproblem in the enumeration approach. In particular, we consider the previous problem with $N = 50$, $n = 5$, $m = 80$, $s = 7$. Here we set $\tau_0 = 0.1$, $\alpha_\tau = 1.5$, $\epsilon_{\text{in}} = 0.02$ for PD and $\tau_0 = 1$, $\alpha_\tau = 1.5$ $m = 4$, $\sigma = 0.05$, $\epsilon_{\text{in}} = 0.1$ for the ALM. The experiment is repeated 20 times for different random seeds. The results are reported in the form of performance profiles in Fig. 3.

We deduce that Penalty Decomposition can indeed be a good choice in particularly complicated settings: the global optimizer was always reached, and this result was obtained in a consistently more efficient way than the brute force approach; the ALM does not have a comparable appeal in this context, the reason arguably being the much higher frequency of it resorting to the projection operation.

Table 7 Results of experiments on disjoint programming problems, for increasing number of feasible set components N

N	Algorithm	f_val	Runtime (s)	n_proj
2	Enumeration + SLSQP	140.58	0.61	–
	PD	140.58	2.10	65
	ALM	140.64	12.02	961
5	Enumeration + SLSQP	139.40	2.37	–
	PD	139.40	6.92	93
	ALM	139.59	67.22	713
10	Enumeration + SLSQP	135.82	1.87	–
	PD	135.82	5.87	99
	ALM	135.83	48.13	579
20	Enumeration + SLSQP	134.94	9.96	–
	PD	134.94	19.41	110
	ALM	134.95	170.31	1138
50	Enumeration + SLSQP	135.08	18.44	–
	PD	135.08	41.66	74
	ALM	135.08	267.95	560
	Enumeration + SLSQP	135.69	30.03	–
	PD	135.69	61.26	124
	ALM	135.69	465.91	1039
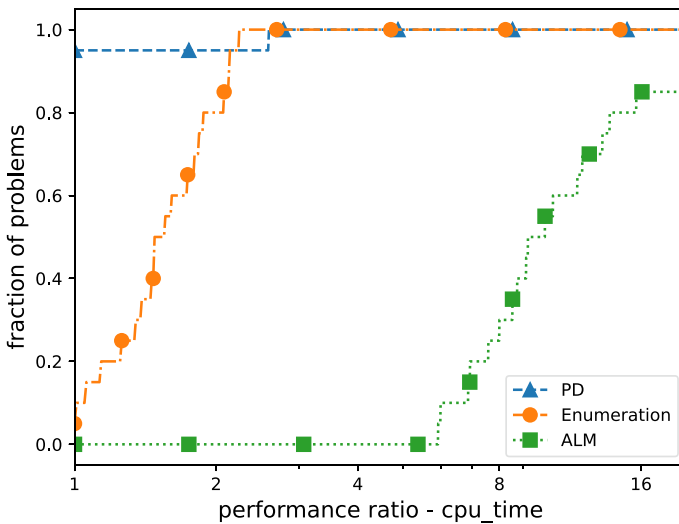
Fig. 3 Performance profiles of runtime attained by PD, ALM and the enumeration strategy (with SLSQP) on 20 disjoint programming problems. When a solver does not attain the global minimum, the corresponding runtime is considered infinite when building the profile

7 Conclusions

The current paper considers a penalty decomposition scheme for optimization problems with geometric constraints. It generalizes existing penalty decomposition schemes both by taking advantage of a general abstract (and usually complicated) constraint (as opposed to having only particular instances like cardinality constraints) and by including further (though supposingly simple) constraints. The idea and the convergence theory of this method is also related to recent augmented Lagrangian techniques, but the decomposition idea turns out to be numerically superior by allowing more efficient subproblem solvers and using many less projection steps.

In principle, it should be possible to extend the decomposition idea to the class of (safeguarded) augmented Lagrangian methods. Another, and related, question is whether one can exploit additional properties of augmented Lagrangian methods in order to improve the existing convergence theory. For example, augmented Lagrangian techniques have very strong convergence properties in the convex case. The particular classes of problems discussed in this paper are nonconvex, but the nonconvexity mainly arises from the fact that the abstract set D is nonconvex. Since we deal with the complicated set D explicitly, so that all iterates are feasible with respect to this set, a natural question is therefore whether one can prove stronger convergence properties in those situations where the remaining functions and constraints are convex. This will be part of our future research.

Data availability Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Code availability All the code developed for the experimental part of this paper is publicly available at https://github.com/MatteoLapucci/GeoIPD.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Jia, X., Kanzow, C., Mehlitz, P., Wachsmuth, G.: An augmented lagrangian method for optimization problems with structured geometric constraints. Progr. Math. (2022). https://doi.org/10.1007/s10107-022-01870-z

2. Benko, M., Červinka, M., Hoheisel, T.: Sufficient conditions for metric subregularity of constraint systems with applications to disjunctive and ortho-disjunctive programs. Set-Val. Variat. Anal. **30**(1), 143–177 (2022)

3. Benko, M., Gfrerer, H.: New verifiable stationarity concepts for a class of mathematical programs with disjunctive constraints. Optimization **67**(1), 1–23 (2018)

4. Flegel, M.L., Kanzow, C., Outrata, J.V.: Optimality conditions for disjunctive programs with application to mathematical programs with equilibrium constraints. Set-Val. Anal. **15**(2), 139–162 (2007)

5. Mehlitz, P.: On the linear independence constraint qualification in disjunctive programming. Optimization **69**(10), 2241–2277 (2020)

6. Ye, J.: Optimality conditions for optimization problems with complementarity constraints. SIAM J. Optimiz. **9**(2), 374–387 (1999)

7. Achtziger, W., Kanzow, C.: Mathematical programs with vanishing constraints: optimality conditions and constraint qualifications. Math. Progr. **114**(1), 69–99 (2008)

8. Mehlitz, P.: Stationarity conditions and constraint qualifications for mathematical programs with switching constraints. Math. Progr. **181**(1), 149–186 (2020)

9. Lapucci, M.: Theory and algorithms for sparsity constrained optimization problems. PhD thesis, University of Florence, Italy (2022)

10. Lapucci, M., Levato, T., Sciandrone, M.: Convergent inexact penalty decomposition methods for cardinality-constrained problems. J. Optimiz. Theory Appl. **188**(2), 473–496 (2021)

11. Kishore Kumar, N., Schneider, J.: Literature survey on low rank approximation of matrices. Linear Multilin. Algebra **65**(11), 2212–2244 (2017)

12. Markovsky, I.: Low rank approximation: algorithms, implementation, applications, 2nd edn. Springer, London, UK (2012)

13. Galvan, G., Lapucci, M., Levato, T., Sciandrone, M.: An alternating augmented Lagrangian method for constrained nonconvex optimization. Optimiz. Method. Softw. **35**(3), 502–520 (2020)

14. Lu, Z., Zhang, Y.: Sparse approximation via penalty decomposition methods. SIAM J. Optimiz. **23**(4), 2448–2478 (2013)

15. Zhang, Y., Lu, Z.: Penalty decomposition methods for rank minimization. Adv. Neural Inf. Process. Sys. **24** (2011)

16. Guignard, M., Kim, S.: Lagrangean decomposition: a model yielding stronger Lagrangean bounds. Math. Progr. **39**(2), 215–228 (1987)

17. Jörnsten, K.O., Näsberg, M., Smeds, P.A.: Variable splitting: a new Lagrangean relaxation approach to some mathematical programming models. Universitetet i Linköping/Tekniska Högskolan i Linköping, Department of Mathematics (1985)

18. Grippo, L., Sciandrone, M.: Globally convergent block-coordinate techniques for unconstrained optimization. Optimiz. Meth. Softw. **10**(4), 587–637 (1999)

19. Bonettini, S.: Inexact block coordinate descent methods with application to non-negative matrix factorization. IMA J. Numer. Anal. **31**(4), 1431–1452 (2011)

20. Bauschke, H.H., Combettes, P.L.: Convex analysis and monotone operator theory in hilbert spaces, 1st edn. Springer, New York (2011). https://doi.org/10.1007/978-1-4419-9467-7

21. Mordukhovich, B.S.: Variational analysis and applications, 1st edn. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92775-6

22. Mehlitz, P.: Asymptotic stationarity and regularity for nonsmooth optimization problems. J Nonsmooth Anal. Optimiz. **1** (2020)

23. Andreani, R., Haeser, G., Martínez, J.M.: On sequential optimality conditions for smooth constrained optimization. Optimization **60**(5), 627–641 (2011)

24. Andreani, R., Martinez, J.M., Ramos, A., Silva, P.J.: A cone-continuity constraint qualification and algorithmic consequences. SIAM J. Optimiz. **26**(1), 96–110 (2016)

25. Andreani, R., Martinez, J.M., Ramos, A., Silva, P.J.: Strict constraint qualifications and sequential optimality conditions for constrained optimization. Math. Operat. Res. **43**(3), 693–717 (2018)

26. Rockafellar, R.T., Wets, R.J.-B.: Variational analysis, 1st Edn. Springer, Heidelberg(2009) https://doi.org/10.1007/978-3-642-02431-3

27. Börgens, E., Kanzow, C., Steck, D.: Local and global analysis of multiplier methods for constrained optimization in Banach spaces. SIAM J. Contr. Optimiz. **57**(6), 3694–3722 (2019)

28. Kanzow, C., Steck, D.: An example comparing the standard and safeguarded augmented Lagrangian methods. Operat. Res. Lett. **45**(6), 598–603 (2017)

29. Bertsekas, D.: Nonlinear programming, vol. 4, 2nd edn. Athena Scientific, Belmont (2016)

30. Beck, A., Eldar, Y.C.: Sparsity constrained nonlinear optimization: optimality conditions and algorithms. SIAM J. Optimiz. **23**(3), 1480–1509 (2013). https://doi.org/10.1137/120869778
31. Lämmel, S., Shikhman, V.: On nondegenerate M-stationary points for sparsity constrained nonlinear optimization. J. Global Optimiz. **82**(2), 219–242 (2022)
32. Ben-Tal, A., Nemirovski, A.: Lectures on modern convex optimization: analysis, algorithms, and engineering applications, 1st edn. SIAM, Philadelphia (2001)
33. Burer, S., Monteiro, R.D., Zhang, Y.: Maximum stable set formulations and heuristics based on continuous optimization. Math. Progr. **94**(1), 137–166 (2002)
34. Candès, E.J., Recht, B.: Exact matrix completion via convex optimization. Foundat. Comp. Math. **9**(6), 717–772 (2009)
35. Recht, B., Fazel, M., Parrilo, P.A.: Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM Review **52**(3), 471–501 (2010)
36. Hosseini, S., Luke, D.R., Uschmajew, A.: Tangent and normal cones for low-rank matrices. In: Nonsmooth optimization and its applications, pp. 45–53. Springer, Birkhäuser, Cham (2019)
37. Burdakov, O.P., Kanzow, C., Schwartz, A.: Mathematical programs with cardinality constraints: reformulation by complementarity-type conditions and a regularization method. SIAM J. Optimiz. **26**(1), 397–425 (2016)
38. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Math. Progr. **45**(1), 503–528 (1989)
39. Bertsimas, D., Cory-Wright, R.: A scalable algorithm for sparse portfolio selection. Informs J. Comput. **34**(3), 1489–1511 (2022)
40. Gurobi optimization, LLC: Gurobi optimizer reference manual (2022). https://www.gurobi.com
41. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Progr. **91**(2), 201–213 (2002)
42. Cocchi, G., Levato, T., Liuzzi, G., Sciandrone, M.: A concave optimization-based approach for sparse multiobjective programming. Optimiz. Lett. **14**(3), 535–556 (2020)
43. Zhang, Y., Yang, Q.: A survey on multi-task learning. IEEE Trans. Knowl. Data Eng. **34**(12), 5586–5609 (2021)
44. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: data mining, inference, and prediction, 2nd edn. Springer, New York (2009)
45. Xue, Y., Liao, X., Carin, L., Krishnapuram, B.: Multi-task learning for classification with Dirichlet process priors. J. Mach. Learn. Res. **8**(1), 35–63 (2007)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.