An Inexact Augmented Lagrangian Framework for Nonconvex Optimization with Nonlinear Constraints

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

We propose a practical inexact augmented Lagrangian method (iALM) for nonconvex problems with nonlinear constraints. We characterize the total computational complexity of our method subject to a verifiable geometric condition, which is closely related to the Polyak-Lojasiewicz and Mangasarian-Fromowitz conditions.

In particular, when a first-order solver is used for the inner iterates, we prove that iALM finds a first-order stationary point with $\tilde{\mathcal{O}}(1/\epsilon^3)$ calls to the first-order oracle. If, in addition, the problem is smooth and a second-order solver is used for the inner iterates, iALM finds a second-order stationary point with $\tilde{\mathcal{O}}(1/\epsilon^5)$ calls to the second-order oracle. These complexity results match the known theoretical results in the literature.

We also provide strong numerical evidence on large-scale machine learning problems, including the Burer-Monteiro factorization of semidefinite programs, and a novel nonconvex relaxation of the standard basis pursuit template. For these examples, we also show how to verify our geometric condition.

1 Introduction

We study the nonconvex optimization problem

$$\min_{x\in\mathbb{R}^d} f(x)+g(x) \quad \text{s.t.} \quad A(x)=0, \tag{1}$$

where $f:\mathbb{R}^d\to\mathbb{R}$ is a continuously-differentiable nonconvex function and $A:\mathbb{R}^d\to\mathbb{R}^m$ is a nonlinear operator. We assume that $g:\mathbb{R}^d\to\mathbb{R}$ is a proximal-friendly convex function [47].

A host of problems in computer science [33, 37, 69], machine learning [40, 58], and signal processing [56, 57] naturally fall under the template (1), including max-cut, clustering, generalized eigenvalue decomposition, as well as the quadratic assignment problem (QAP) [69].

To solve (1), we propose an intuitive and easy-to-implement augmented Lagrangian algorithm, and provide its total iteration complexity under an interpretable geometric condition. Before we elaborate on the results, let us first motivate (1) with an application to semidefinite programming (SDP):

Vignette: Burer-Monteiro splitting. A powerful convex relaxation for max-cut, clustering, and many others is provided by the SDP

$$\min_{X\in\mathbb{S}^{d\times d}} \langle C, X\rangle \quad \text{s.t.} \quad B(X)=b,\ X\succeq 0, \tag{2}$$

where $C\in\mathbb{R}^{d\times d}$, X is a positive semidefinite $d\times d$ matrix, and $B:\mathbb{S}^{d\times d}\to\mathbb{R}^m$ is a linear operator. If the unique-games conjecture is true, the SDP (2) obtains the best possible approximation for the underlying discrete problem [53].

Since d is often large, many first- and second-order methods for solving such SDP's are immediately ruled out, not only due to their high computational complexity, but also due to their storage requirements, which are $\mathcal{O}(d^2)$.

A contemporary challenge in optimization is therefore to solve SDPs using little space and in a scalable fashion. The recent homotopy conditional gradient method, which is based on linear minimization oracles (LMOs), can solve (2) in a small space via sketching [68]. However, such LMO-based methods are extremely slow in obtaining accurate solutions.

A different approach for solving (1), dating back to [14, 15], is the so-called Burer-Monteiro (BM) factorization $X = UU^\top$, where $U \in \mathbb{R}^{d\times r}$ and r is selected according to the guidelines in [49, 1], which is tight [62]. The BM factorization leads to the following nonconvex problem in the template (1):

$$\min_{U\in\mathbb{R}^{d\times r}} \langle C, UU^\top\rangle \quad \text{s.t.} \quad B(UU^\top) = b, \tag{3}$$

The BM factorization does not introduce any extraneous local minima [15]. Moreover, [13] established the connection between the local minimizers of the factorized problem (3) and the global minimizers for (2). To solve (3), the inexact Augmented Lagrangian method (iALM) is widely used [14, 15, 35], due to its cheap per iteration cost and its empirical success.

Every (outer) iteration of iALM calls a solver to solve an intermediate augmented Lagrangian subproblem to near stationarity. The choices include first-order methods, such as the proximal gradient descent [47], or second-order methods, such as the trust region method and BFGS [44].[1]

Unlike its convex counterpart [41, 36, 64], the convergence rate and the complexity of iALM for (3) are not well-understood, see Section 5 for a review of the related literature. Indeed, addressing this important theoretical gap is one of the contributions of our work. In addition

▷ We derive the convergence rate of iALM for solving (1) to first- or second-order optimality, and find the total iteration complexity of iALM using different solvers for the augmented Lagrangian subproblems. Our complexity bounds match the best theoretical results in optimization, see Section 5.

▷ Our iALM framework is future-proof in the sense that different subsolvers can be substituted.

▷ We propose a geometric condition that simplifies the algorithmic analysis for iALM, and clarify its connection to well-known Polyak-Lojasiewicz [32] and Mangasarian-Fromovitz [3] conditions. We also verify this condition for key problems in Section 6.

2 Preliminaries

Notation. We use the notation $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$ for the standard inner product and the norm on $\mathbb{R}^d$. For matrices, $\|\cdot\|$ and $\|\cdot\|_F$ denote the spectral and the Frobenius norms, respectively. For the convex function $g : \mathbb{R}^d \to \mathbb{R}$, the subdifferential set at $x \in \mathbb{R}^d$ is denoted by $\partial g(x)$ and we will occasionally use the notation $\partial g(x)/\beta = \{z/\beta : z \in \partial g(x)\}$. When presenting iteration complexity results, we often use $\widetilde{O}(\cdot)$ which suppresses the logarithmic dependencies.

We denote $\delta_{\mathcal{X}} : \mathbb{R}^d \to \mathbb{R}$ as the indicator function of a set $\mathcal{X} \subset \mathbb{R}^d$. The distance function from a point x to $\mathcal{X}$ is denoted by $\text{dist}(x, \mathcal{X}) = \min_{z\in\mathcal{X}} \|x - z\|$. For integers $k_0 \le k_1$, we use the notation $[k_0 : k_1] = \{k_0, \ldots, k_1\}$. For an operator $A : \mathbb{R}^d \to \mathbb{R}^m$ with components $\{A_i\}_{i=1}^m$, $DA(x) \in \mathbb{R}^{m\times d}$ denotes the Jacobian of A, where the ith row of $DA(x)$ is the vector $\nabla A_i(x) \in \mathbb{R}^d$.

Smoothness. We assume smooth $f : \mathbb{R}^d \to \mathbb{R}$ and $A : \mathbb{R}^d \to \mathbb{R}^m$; i.e., there exist $\lambda_f, \lambda_A \ge 0$ s.t.

$$\|\nabla f(x) - \nabla f(x')\| \le \lambda_f \|x - x'\|, \quad \|DA(x) - DA(x')\| \le \lambda_A \|x - x'\|, \quad \forall x, x' \in \mathbb{R}^d. \tag{4}$$

Augmented Lagrangian method (ALM). ALM is a classical algorithm, which first appeared in [29, 51] and extensively studied afterwards in [3, 8]. For solving (1), ALM suggests solving the problem

$$\min_x \max_y \; \mathcal{L}_\beta(x, y) + g(x), \tag{5}$$

where, for penalty weight $\beta > 0$, $\mathcal{L}_\beta$ is the corresponding augmented Lagrangian, defined as

$$\mathcal{L}_\beta(x, y) := f(x) + \langle A(x), y\rangle + \frac{\beta}{2}\|A(x)\|^2. \tag{6}$$

[1]Strictly speaking, BFGS is in fact a quasi-Newton method that emulates second-order information.

The minimax formulation in (5) naturally suggests the following algorithm for solving (1):

$$x_{k+1} \in \underset{x}{\operatorname{argmin}} \ \mathcal{L}_\beta(x, y_k) + g(x), \tag{7}$$

$$y_{k+1} = y_k + \sigma_k A(x_{k+1}),$$

where the dual step sizes are denoted as $\{\sigma_k\}_k$. However, computing x_{k+1} above requires solving the nonconvex problem (7) to optimality, which is typically intractable. Instead, it is often easier to find an approximate first- or second-order stationary point of (7).

Hence, we argue that by gradually improving the stationarity precision and increasing the penalty weight β above, we can reach a stationary point of the main problem in (1), as detailed in Section 3.

Optimality conditions. First-order necessary optimality conditions for (1) are well-studied. Indeed, $x \in \mathbb{R}^d$ is a first-order stationary point of (1) if there exists $y \in \mathbb{R}^m$ such that

$$-\nabla_x \mathcal{L}_\beta(x, y) \in \partial g(x), \qquad A(x) = 0, \tag{8}$$

which is in turn the necessary optimality condition for (5). Inspired by this, we say that x is an (ϵ_f, β) first-order stationary point of (5) if there exists a $y \in \mathbb{R}^m$ such that

$$\operatorname{dist}(-\nabla_x \mathcal{L}_\beta(x, y), \partial g(x)) \le \epsilon_f, \qquad \|A(x)\| \le \epsilon_f, \tag{9}$$

for $\epsilon_f \ge 0$. In light of (9), a metric for evaluating the stationarity of a pair $(x, y) \in \mathbb{R}^d \times \mathbb{R}^m$ is

$$\operatorname{dist}\left(-\nabla_x \mathcal{L}_\beta(x, y), \partial g(x)\right) + \|A(x)\|, \tag{10}$$

which we use as the first-order stopping criterion. As an example, for a convex set $\mathcal{X} \subset \mathbb{R}^d$, suppose that $g = \delta_\mathcal{X}$ is the indicator function on $\mathcal{X}$. Let also $T_\mathcal{X}(x) \subseteq \mathbb{R}^d$ denote the tangent cone to $\mathcal{X}$ at x, and with $P_{T_\mathcal{X}(x)} : \mathbb{R}^d \to \mathbb{R}^d$ we denote the orthogonal projection onto this tangent cone. Then, for $u \in \mathbb{R}^d$, it is not difficult to verify that

$$\operatorname{dist}\left(u, \partial g(x)\right) = \|P_{T_\mathcal{X}(x)}(u)\|. \tag{11}$$

When $g = 0$, a first-order stationary point $x \in \mathbb{R}^d$ of (1) is also second-order stationary if

$$\lambda_{\text{min}}(\nabla_{xx} \mathcal{L}_\beta(x, y)) \ge 0, \tag{12}$$

where $\nabla_{xx}\mathcal{L}_\beta$ is the Hessian of $\mathcal{L}_\beta$ with respect to x, and $\lambda_{\text{min}}(\cdot)$ returns the smallest eigenvalue of its argument. Analogously, x is an $(\epsilon_f, \epsilon_s, \beta)$ second-order stationary point if, in addition to (**??**), it holds that

$$\lambda_{\text{min}}(\nabla_{xx} \mathcal{L}_\beta(x, y)) \ge -\epsilon_s, \tag{13}$$

for $\epsilon_s \ge 0$. Naturally, for second-order stationarity, we use $\lambda_{\text{min}}(\nabla_{xx}\mathcal{L}_\beta(x, y))$ as the stopping criterion.

Smoothness lemma. This next result controls the smoothness of $\mathcal{L}_\beta(\cdot, y)$ for a fixed y. The proof is standard but nevertheless is included in Appendix C for completeness.

Lemma 2.1 (smoothness). *For fixed $y \in \mathbb{R}^m$ and $\rho, \rho' \ge 0$, it holds that*

$$\|\nabla_x \mathcal{L}_\beta(x, y) - \nabla_x \mathcal{L}_\beta(x', y)\| \le \lambda_\beta \|x - x'\|, \tag{14}$$

for every $x, x' \in \{x'' : \|x''\| \le \rho, \|A(x'')\| \le \rho'\}$, where

$$\lambda_\beta \le \lambda_f + \sqrt{m}\lambda_A \|y\| + (\sqrt{m}\lambda_A \rho' + d\lambda_A'^2)\beta =: \lambda_f + \sqrt{m}\lambda_A\|y\| + \lambda''(A, \rho, \rho')\beta. \tag{15}$$

Above, λ_f, λ_A were defined in (4) and

$$\lambda'_A := \max_{\|x\| \le \rho} \|DA(x)\|. \tag{16}$$

3 Algorithm

To solve the equivalent formulation of (1) presented in (5), we propose the inexact ALM (iALM), detailed in Algorithm 1. At the kth iteration, Step 2 of Algorithm 1 calls a solver that finds an approximate stationary point of the augmented Lagrangian $\mathcal{L}_{\beta_k}(\cdot, y_k)$ with the accuracy of ϵ_{k+1}, and this accuracy gradually increases in a controlled fashion. The increasing sequence of penalty weights $\{\beta_k\}_k$ and the dual update (Steps 4 and 5) are responsible for continuously enforcing the constraints in (1). The appropriate choice for $\{\beta_k\}_k$ will be specified in Corrollary Sections A.1 and A.2.

The particular choice of the dual step sizes $\{\sigma_k\}_k$ in Algorithm 1 ensures that the dual variable y_k remains bounded, see [2] in the ALM literature where a similar dual step size is considered.

Algorithm 1 Inexact ALM for solving (1)

```
Input: Non-decreasing, positive, unbounded sequence ${\beta_k}_{k\geq 1}$, stopping thresholds $\tau_f, \tau_s > 0$.
Initialization: Primal variable $x_1 \in \mathbb{R}^d$, dual variable $y_0 \in \mathbb{R}^m$, dual step size $\sigma_1 > 0$.
for $k = 1, 2, \ldots$ do
    1. (Update tolerance) $\epsilon_{k+1} = 1/\beta_k$.
    2. (Inexact primal solution) Obtain $x_{k+1} \in \mathbb{R}^d$ such that
           $\mathrm{dist}(-\nabla_x \mathcal{L}_{\beta_k}(x_{k+1}, y_k), \partial g(x_{k+1})) \leq \epsilon_{k+1}$
       for first-order stationarity
           $\lambda_{\text{min}}(\nabla_{xx}\mathcal{L}_{\beta_k}(x_{k+1}, y_k)) \geq -\epsilon_{k+1}$
       for second-order-stationarity, if $g = 0$ in (1).
    3. (Update dual step size)
           $\sigma_{k+1} = \sigma_1 \min\Big(\frac{\|A(x_1)\| \log^2 2}{\|A(x_{k+1})\|(k+1)\log^2(k+2)}, 1\Big)$.
    4. (Dual ascent) $y_{k+1} = y_k + \sigma_{k+1} A(x_{k+1})$.
    5. (Stopping criterion) If
           $\mathrm{dist}(-\nabla_x \mathcal{L}_{\beta_k}(x_{k+1}), \partial g(x_{k+1})) + \|A(x_{k+1})\| \leq \tau_f$,
       for first-order stationarity and if also $\lambda_{\text{min}}(\nabla_{xx}\mathcal{L}_{\beta_k}(x_{k+1}, y_k)) \geq -\tau_s$ for second-order
       stationarity, then quit and return $x_{k+1}$ as an (approximate) stationary point.
end for
```

4 Convergence Rate

This section presents the total iteration complexity of Algorithm 1 for finding first and second-order stationary points of problem (1). All the proofs are deferred to Appendix B. Theorem 4.1 characterizes the convergence rate of Algorithm 1 for finding stationary points in the number of outer iterations.

Theorem 4.1. **(convergence rate)** *For integers $2 \leq k_0 \leq k_1$, consider the interval $K = [k_0 : k_1]$, and let $\{x_k\}_{k\in K}$ be the output sequence of Algorithm 1 on the interval K.*[2] *Let also $\rho := \sup_{k\in[K]} \|x_k\|$.*[3] *Suppose that f and A satisfy (4) and let*

$$\lambda'_f = \max_{\|x\|\leq\rho} \|\nabla f(x)\|, \qquad \lambda'_A = \max_{\|x\|\leq\rho} \|DA(x)\|, \tag{17}$$

be the (restricted) Lipschitz constants of f and A, respectively. With $\nu > 0$, assume that

$$\nu\|A(x_k)\| \leq \mathrm{dist}\left(-DA(x_k)^\top A(x_k), \frac{\partial g(x_k)}{\beta_{k-1}}\right), \tag{18}$$

for every $k \in K$. We consider two cases:

- *If a first-order solver is used in Step 2, then x_k is an $(\epsilon_{k,f}, \beta_k)$ first-order stationary point of (1) with*

$$\epsilon_{k,f} = \frac{1}{\beta_{k-1}}\left(\frac{2(\lambda'_f + \lambda'_A y_{\max})(1 + \lambda'_A \sigma_k)}{\nu} + 1\right) =: \frac{Q(f, g, A, \sigma_1)}{\beta_{k-1}}, \tag{19}$$

  for every $k \in K$, where $y_{\max}(x_1, y_0, \sigma_1)$ is specified in (40) due to the limited space.

- *If a second-order solver is used in Step 2, then x_k is an $(\epsilon_{k,f}, \epsilon_{k,s}, \beta_k)$ second-order stationary point of (1) with $\epsilon_{k,s}$ specified above and with*

$$\epsilon_{k,s} = \epsilon_{k-1} + \sigma_k\sqrt{m}\lambda_A \frac{2\lambda'_f + 2\lambda'_A y_{\max}}{\nu\beta_{k-1}} = \frac{\nu + \sigma_k\sqrt{m}\lambda_A 2\lambda'_f + 2\lambda'_A y_{\max}}{\nu\beta_{k-1}} =: \frac{Q'(f, g, A, \sigma_1)}{\beta_{k-1}}. \tag{20}$$

[2]The choice of $k_1 = \infty$ is valid here too.

[3]If necessary, to ensure that $\rho < \infty$, one can add a small factor of $\|x\|^2$ to $\mathcal{L}_\beta$ in (6). Then it is easy to verify that the iterates of Algorithm 1 remain bounded, provided that the initial penalty weight β_0 is large enough, $\sup_x \|\nabla f(x)\|/\|x\| < \infty$, $\sup_x \|A(x)\| < \infty$, and $\sup_x \|DA(x)\| < \infty$.

Theorem 4.1 states that Algorithm 1 converges to a (first- or second-) order stationary point of (1) at the rate of $1/\beta_k$, further specified in Corollary 4.2 and Corollary 4.3. A few remarks are in order about Theorem 4.1.

Regularity. The key geometric condition in Theorem 4.1 is (18) which, broadly speaking, ensures that the primal updates of Algorithm 1 reduce the feasibility gap as the penalty weight β_k grows. We will verify this condition for several examples in Section 6.

This condition in (18) is closely related to those in the existing literature. In the special case where $g = 0$ in (1), it is easy to verify that (18) reduces to the Polyak-Lojasiewicz (PL) condition for minimizing $\|A(x)\|^2$ [32]. PL condition itself is a special case of Kurdyka-Lojasiewicz with $\theta = 1/2$, see [65, Definition 1.1]. When $g = 0$, it is also easy to see that (18) is weaker than the Mangasarian-Fromovitz (MF) condition in nonlinear optimization [10, Assumption 1]. Moreover, when g is the indicator on a convex set, (18) is a consequence of the *basic constraint qualification* in [54], which itself generalizes the MF condition to the case when g is an indicator function of a convex set.

We may think of (18) as a local condition, which should hold within a neighborhood of the constraint set $\{x : A(x) = 0\}$ rather than everywhere in $\mathbb{R}^d$. There is a constant complexity algorithm in [10] to reach this so-called "information zone", which supplements Theorem 4.1. Lastly, in contrast to most conditions in the nonconvex optimization literature, such as [25], the condition in (18) appears to be easier to verify, as we see in the sequel.

Penalty method. A classical algorithm to solve (1) is the penalty method, which is characterized by the absence of the dual variable ($y = 0$) in (6). Indeed, ALM can be interpreted as an adaptive penalty or smoothing method with a variable center determined by the dual variable. It is worth noting that, with the same proof technique, one can establish the same convergence rate of Theorem 4.1 for the penalty method. However, while both methods have the same convergence rate in theory, we ignore the uncompetitive penalty method since it is significantly outperformed by iALM in practice.

Computational complexity. Theorem 4.1 specifies the number of (outer) iterations that Algorithm 1 requires to reach a near-stationary point of problem (6) with a prescribed precision and, in particular, specifies the number of calls made to the solver in Step 2. In this sense, Theorem 4.1 does not fully capture the computational complexity of Algorithm 1, as it does not take into account the computational cost of the solver in Step 2.

To better understand the total iteration complexity of Algorithm 1, we consider two scenarios in the following. In the first scenario, we take the solver in Step 2 to be the Accelerated Proximal Gradient Method (APGM), a well-known first-order algorithm [27]. In the second scenario, we will use the second-order trust region method developed in [17]. We have the following two corollaries showing the total complexity of our algorithm to reach first and second-order stationary points. Appendix **??** contains the proofs and more detailed discussion for the complexity results.

Corollary 4.2 (First-order optimality)**.** *For $b > 1$, let $\beta_k = b^k$ for every k. If we use APGM from [27] for Step 2 of Algorithm 1, the algorithm finds an (ϵ_f, β_k) first-order stationary point, after T calls to the first-order oracle, where*

$$T = \mathcal{O}\left(\frac{Q^3\rho^2}{\epsilon^3}\log_b\left(\frac{Q}{\epsilon}\right)\right) = \tilde{\mathcal{O}}\left(\frac{Q^3\rho^2}{\epsilon^3}\right). \tag{21}$$

For Algorithm 1 to reach a near-stationary point with an accuracy of ϵ_f in the sense of (**??**) and with the lowest computational cost, we therefore need to perform only one iteration of Algorithm 1, with β_1 specified as a function of ϵ_f by (19) in Theorem 4.1. In general, however, the constants in (19) are unknown and this approach is thus not feasible. Instead, the homotopy approach taken by Algorithm 1 ensures achieving the desired accuracy by gradually increasing the penalty weight. This homotopy approach increases the computational cost of Algorithm 1 only by a factor logarithmic in the ϵ_f, as detailed in the proof of Corollary 4.2.

Corollary 4.3 (Second-order optimality)**.** *For $b > 1$, let $\beta_k = b^k$ for every k. We assume that*

$$\mathcal{L}_\beta(x_1, y) - \min_x \mathcal{L}_\beta(x, y) \le L_u, \qquad \forall \beta. \tag{22}$$

If we use the trust region method from [17] for Step 2 of Algorithm 1, the algorithm finds an ϵ-second-order stationary point of (1) in T calls to the second-order oracle where

$$T = \mathcal{O}\left(\frac{L_u Q'^5}{\epsilon^5}\log_b\left(\frac{Q'}{\epsilon}\right)\right) = \widetilde{\mathcal{O}}\left(\frac{L_u Q'^5}{\epsilon^5}\right). \tag{23}$$

Remark. These complexity results for first and second-order are stationarity with respect to (6). We note that these complexities match [18] and [7]. However, the stationarity criteria and the definition of dual variable in these papers differ from ours. We include more discussion on this in the Appendix.

5 Related Work

ALM has a long history in the optimization literature, dating back to [29, 51]. In the special case of (1) with a convex function f and a linear operator A, standard, inexact, and linearized versions of ALM have been extensively studied [36, 41, 60, 64].

Classical works on ALM focused on the general template of (1) with nonconvex f and nonlinear A, with arguably stronger assumptions and required exact solutions to the subproblems of the form (7), which appear in Step 2 of Algorithm 1, see for instance [4].

A similar analysis was conducted in [22] for the general template of (1). The authors considered inexact ALM and proved convergence rates for the outer iterates, under specific assumptions on the initialization of the dual variable. However, in contrast, the authors did not analyze how to solve the subproblems inexactly and did not provide total complexity results with verifiable conditions.

Problem (1) with similar assumptions to us is also studied in [7] and [18] for first-order and second-order stationarity, respectively, with explicit iteration complexity analysis. As we have mentioned in Section 4, our iteration complexity results matches these theoretical algorithms with a simpler algorithm and a simpler analysis. In addition, these algorithms require setting final accuracies since they utilize this information in the algorithm while our Algorithm 1 does not set accuracies a priori.

[16] also considers the same template (1) for first-order stationarity with a penalty-type method instead of ALM. Even though the authors show $\mathcal{O}(1/\epsilon^2)$ complexity, this result is obtained by assuming that the penalty parameter remains bounded. We note that such an assumption can also be used to improve our complexity results to match theirs.

[10] studies the general template (1) with specific assumptions involving local error bound conditions for the (1). These conditions are studied in detail in [9], but their validity for general SDPs (2) has never been established. This work also lacks the total iteration complexity analysis presented here.

Another work [20] focused on solving (1) by adapting the primal-dual method of Chambolle and Pock [19]. The authors proved the convergence of the method and provided convergence rate by imposing error bound conditions on the objective function that do not hold for standard SDPs.

[14, 15] is the first work that proposes the splitting $X = UU^\top$ for solving SDPs of the form (2). Following these works, the literature on Burer-Monteiro (BM) splitting for the large part focused on using ALM for solving the reformulated problem (3).

However, this proposal has a few drawbacks: First, it requires exact solutions in Step 2 of Algorithm 1 in theory, which in practice is replaced with inexact solutions. Second, their results only establish convergence without providing the rates. In this sense, our work provides a theoretical understanding of the BM splitting with inexact solutions to Step 2 of Algorithm 1 and complete iteration complexities.

[6, 48] are among the earliest efforts to show convergence rates for BM splitting, focusing on the special case of SDPs without any linear constraints. For these specific problems, they prove the convergence of gradient descent to global optima with convergence rates, assuming favorable initialization. These results, however, do not apply to general SDPs of the form (2) where the difficulty arises due to the linear constraints.

Another popular method for solving SDPs are due to [12, 11, 13], focusing on the case where the constraints in (1) can be written as a Riemannian manifold after BM splitting. In this case, the authors apply the Riemannian gradient descent and Riemannian trust region methods for obtaining first- and second-order stationary points, respectively. They obtain $\mathcal{O}(1/\epsilon^2)$ complexity for finding first-order stationary points and $\mathcal{O}(1/\epsilon^3)$ complexity for finding second-order stationary points.

While these complexities appear better than ours, the smooth manifold requirement in these works is indeed restrictive. In particular, this requirement holds for max-cut and generalized eigenvalue problems, but it is not satisfied for other important SDPs such as quadratic programming (QAP), optimal power flow and clustering with general affine constraints. In addition, as noted in [11], per iteration cost of their method for max-cut problem is an astronomical $\mathcal{O}(d^6)$.

Lastly, there also exists a line of work for solving SDPs in their original convex formulation, in a storage efficient way [42, 67, 68]. These works have global optimality guarantees by their virtue of directly solving the convex formulation. On the downside, these works require the use of eigenvalue routines and exhibit significantly slower convergence as compared to nonconvex approaches [31].

6 Numerical Evidence

We first begin with a caveat: It is known that quasi-Newton methods, such as BFGS and lBFGS, might not converge for nonconvex problems [21, 38]. For this reason, we have used the trust region method as the second-order solver in our analysis in Section 4, which is well-studied for nonconvex problems [17]. Empirically, however, BFGS and lBGFS are extremely successful and we have therefore opted for those solvers in this section since the subroutine does not affect Theorem 4.1 as long as the subsolver performs well in practice.

6.1 Clustering

Given data points $\{z_i\}_{i=1}^n$, the entries of the corresponding Euclidean distance matrix $D \in \mathbb{R}^{n\times n}$ are $D_{i,j} = \|z_i - z_j\|^2$. Clustering is then the problem of finding a co-association matrix $Y \in \mathbb{R}^{n\times n}$ such that $Y_{ij} = 1$ if points z_i and z_j are within the same cluster and $Y_{ij} = 0$ otherwise. In [50], the authors provide a SDP relaxation of the clustering problem, specified as

$$\min_{Y\in\mathbb{R}^{nxn}} \text{tr}(DY) \quad \text{s.t.} \quad Y\mathbf{1} = \mathbf{1},\ \text{tr}(Y) = s,\ Y \succeq 0,\ Y \geq 0, \tag{24}$$

where s is the number of clusters and Y is both positive semidefinite and has nonnegative entries. Standard SDP solvers do not scale well with the number of data points n, since they often require projection onto the semidefinite cone with the complexity of $\mathcal{O}(n^3)$. We instead use the BM factorization to solve (24), sacrificing convexity to reduce the computational complexity. More specifically, we solve the program

$$\min_{V\in\mathbb{R}^{n\times r}} \text{tr}(DVV^\top) \quad \text{s.t.} \quad VV^\top\mathbf{1} = \mathbf{1},\ \ \|V\|_F^2 \leq s,\ \ V \geq 0, \tag{25}$$

where $\mathbf{1} \in \mathbb{R}^n$ is the vector of all ones. Note that $Y \geq 0$ in (24) is replaced above by the much stronger but easier-to-enforce constraint $V \geq 0$ in (25), see [35] for the reasoning behind this relaxation. Now, we can cast (25) as an instance of (1). Indeed, for every $i \leq n$, let $x_i \in \mathbb{R}^r$ denote the ith row of V. We next form $x \in \mathbb{R}^d$ with $d = nr$ by expanding the factorized variable V, namely, $x := [x_1^\top, \cdots, x_n^\top]^\top \in \mathbb{R}^d$, and then set

$$f(x) = \sum_{i,j=1}^n D_{i,j} \langle x_i, x_j \rangle, \qquad g = \delta_C, \qquad A(x) = [x_1^\top \sum_{j=1}^n x_j - 1, \cdots, x_n^\top \sum_{j=1}^n x_j - 1]^\top,$$

where C is the intersection of the positive orthant in $\mathbb{R}^d$ with the Euclidean ball of radius $\sqrt{s}$. In Appendix D, we verify that Theorem 4.1 applies to (1) with f, g, A specified above.

In our simulations, we use two different solvers for Step 2 of Algorithm 1, namely, APGM and lBFGS. APGM is a solver for nonconvex problems of the form (7) with convergence guarantees to first-order stationarity, as discussed in Section 4. lBFGS is a limited-memory version of BFGS algorithm in [24] that approximately leverages the second-order information of the problem. We compare our approach against the following convex methods:

- HCGM: Homotopy-based Conditional Gradient Method in [68] which directly solves (24).
- SDPNAL+: A second-order augmented Lagrangian method for solving SDP's with nonnegativity constraints [66].

As for the dataset, our experimental setup is similar to that described by [39]. We use the publicly-available fashion-MNIST data in [63], which is released as a possible replacement for the MNIST handwritten digits. Each data point is a 28×28 gray-scale image, associated with a label from ten classes, labeled from 0 to 9. First, we extract the meaningful features from this dataset using a simple two-layer neural network with a sigmoid activation function. Then, we apply this neural network to 1000 test samples from the same dataset, which gives us a vector of length 10 for each data point,

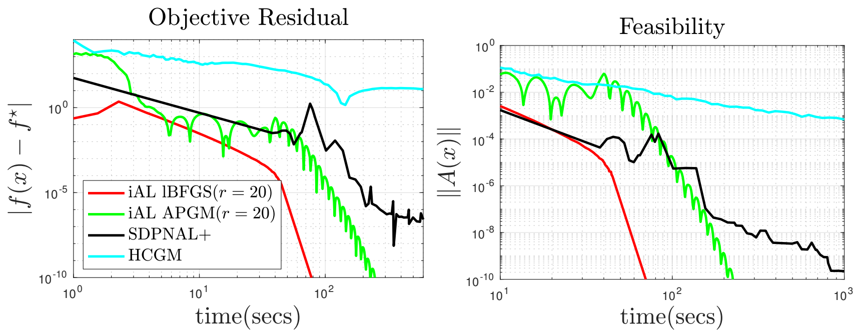

Figure 1: Clustering running time comparison.

where each entry represents the posterior probability for each class. Then, we form the ℓ_2 distance matrix D from these probability vectors. The solution rank for the template (24) is known and it is equal to number of clusters k [35, Theorem 1]. As discussed in [59], setting rank $r > k$ leads more accurate reconstruction in expense of speed. Therefore, we set the rank to 20. The results are depicted in Figure 1. We implemented 3 algorithms on MATLAB and used the software package for SDPNAL+ which contains mex files. It is predictable that the performance of our nonconvex approach would even improve by using mex files.

6.2 Additional demonstrations

We provide several additional experiments in Appendix E. Section E.1 discusses a novel nonconvex relaxation of the standard basis pursuit template which performs comparable to the state of the art convex solvers. In Section E.2, we provide fast numerical solutions to the generalized eigenvalue problem. In Section E.3, we give a contemporary application example that our template applies, namely, denoising with generative adversarial networks. Finally, we provide improved bounds for sparse quadratic assignment problem instances in Section E.4.

7 Conclusions

In this work, we have proposed and analyzed an inexact augmented Lagrangian method for solving nonconvex optimization problems with nonlinear constraints. We prove convergence to the first and second order stationary points of the augmented Lagrangian function, with explicit complexity estimates. Even though the relation of stationary points and global optima is not well-understood in the literature, we find out that the algorithm has fast convergence behavior to either global minima or local minima in a wide variety of numerical experiments.

References

[1] A. I. Barvinok. Problems of distance geometry and convex properties of quadratic maps. *Discrete & Computational Geometry*, 13(2):189–202, 1995.

[2] D. P. Bertsekas. On penalty and multiplier methods for constrained minimization. *SIAM Journal on Control and Optimization*, 14(2):216–235, 1976.

[3] D. P. Bertsekas. Constrained optimization and lagrange multiplier methods. *Computer Science and Applied Mathematics, Boston: Academic Press, 1982*, 1982.

[4] D. P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[5] S. Bhojanapalli, N. Boumal, P. Jain, and P. Netrapalli. Smoothed analysis for low-rank solutions to semidefinite programs in quadratic penalty form. *arXiv preprint arXiv:1803.00186*, 2018.

[6] S. Bhojanapalli, A. Kyrillidis, and S. Sanghavi. Dropping convexity for faster semi-definite optimization. In *Conference on Learning Theory*, pages 530–582, 2016.

[7] E. G. Birgin, J. Gardenghi, J. M. Martinez, S. Santos, and P. L. Toint. Evaluation complexity for nonlinear constrained optimization using unscaled kkt conditions and high-order models. *SIAM Journal on Optimization*, 26(2):951–967, 2016.

[8] E. G. Birgin and J. M. Mart_nez. *Practical augmented Lagrangian methods for constrained optimization*, volume 10. SIAM, 2014.

[9] J. Bolte, T. P. Nguyen, J. Peypouquet, and B. W. Suter. From error bounds to the complexity of first-order descent methods for convex functions. *Mathematical Programming*, 165(2):471–507, 2017.

[10] J. Bolte, S. Sabach, and M. Teboulle. Nonconvex lagrangian-based optimization: monitoring schemes and global convergence. *Mathematics of Operations Research*, 2018.

[11] N. Boumal, P.-A. Absil, and C. Cartis. Global rates of convergence for nonconvex optimization on manifolds. *arXiv preprint arXiv:1605.08101*, 2016.

[12] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a matlab toolbox for optimization on manifolds. *The Journal of Machine Learning Research*, 15(1):1455–1459, 2014.

[13] N. Boumal, V. Voroninski, and A. Bandeira. The non-convex burer-monteiro approach works on smooth semidefinite programs. In *Advances in Neural Information Processing Systems*, pages 2757–2765, 2016.

[14] S. Burer and R. D. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003.

[15] S. Burer and R. D. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3):427–444, 2005.

[16] C. Cartis, N. I. Gould, and P. L. Toint. On the evaluation complexity of composite function minimization with applications to nonconvex nonlinear programming. *SIAM Journal on Optimization*, 21(4):1721–1739, 2011.

[17] C. Cartis, N. I. Gould, and P. L. Toint. Complexity bounds for second-order optimality in unconstrained optimization. *Journal of Complexity*, 28(1):93–108, 2012.

[18] C. Cartis, N. I. Gould, and P. L. Toint. Optimality of orders one to three and beyond: characterization and evaluation complexity in constrained nonconvex optimization. *Journal of Complexity*, 2018.

[19] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40(1):120–145, 2011.

[20] C. Clason, S. Mazurenko, and T. Valkonen. Acceleration and global convergence of a first-order primal–dual method for nonconvex problems. *arXiv preprint arXiv:1802.03347*, 2018.

[21] Y.-H. Dai. Convergence properties of the bfgs algoritm. *SIAM Journal on Optimization*, 13(3):693–701, 2002.

[22] D. Fernandez and M. V. Solodov. Local convergence of exact and inexact augmented lagrangian methods under the second-order sufficient optimality condition. *SIAM Journal on Optimization*, 22(2):384–407, 2012.

[23] J. F. B. Ferreira, Y. Khoo, and A. Singer. Semidefinite programming approach for the quadratic assignment problem with a sparse graph. *Computational Optimization and Applications*, 69(3):677–712, 2018.

[24] R. Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[25] F. Flores-Bazán, F. Flores-Bazán, and C. Vera. A complete characterization of strong duality in nonconvex optimization with a single constraint. *Journal of Global Optimization*, 53(2):185–201, 2012.

[26] R. Ge, C. Jin, P. Netrapalli, A. Sidford, et al. Efficient algorithms for large-scale generalized eigenvector computation and canonical correlation analysis. In *International Conference on Machine Learning*, pages 2741–2750, 2016.

[27] S. Ghadimi and G. Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1-2):59–99, 2016.

[28] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

[29] M. R. Hestenes. Multiplier and gradient methods. *Journal of optimization theory and applications*, 4(5):303–320, 1969.

[30] A. Ilyas, A. Jalal, E. Asteri, C. Daskalakis, and A. G. Dimakis. The Robust Manifold Defense: Adversarial Training using Generative Models. *arXiv e-prints*, page arXiv:1712.09196, Dec. 2017.

[31] M. Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML (1)*, pages 427–435, 2013.

[32] H. Karimi, J. Nutini, and M. Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.

[33] S. Khot and A. Naor. Grothendieck-type inequalities in combinatorial optimization. *arXiv preprint arXiv:1108.2464*, 2011.

[34] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980, Dec. 2014.

[35] B. Kulis, A. C. Surendran, and J. C. Platt. Fast low-rank semidefinite programming for embedding and clustering. In *Artificial Intelligence and Statistics*, pages 235–242, 2007.

[36] G. Lan and R. D. Monteiro. Iteration-complexity of first-order augmented lagrangian methods for convex programming. *Mathematical Programming*, 155(1-2):511–547, 2016.

[37] L. Lovász. Semidefinite programs and combinatorial optimization. In *Recent advances in algorithms and combinatorics*, pages 137–194. Springer, 2003.

[38] W. F. Mascarenhas. The bfgs method with exact line searches fails for non-convex objective functions. *Mathematical Programming*, 99(1):49–61, 2004.

[39] D. G. Mixon, S. Villar, and R. Ward. Clustering subgaussian mixtures by semidefinite programming. *arXiv preprint arXiv:1602.06612*, 2016.

[40] E. Mossel, J. Neeman, and A. Sly. Consistency thresholds for the planted bisection model. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 69–75. ACM, 2015.

[41] V. Nedelcu, I. Necoara, and Q. Tran-Dinh. Computational complexity of inexact gradient augmented lagrangian methods: application to constrained mpc. *SIAM Journal on Control and Optimization*, 52(5):3109–3134, 2014.

[42] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

[43] Y. E. Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. Akad. Nauk SSSR*, volume 269, pages 543–547, 1983.

[44] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[45] M. Nouiehed, J. D. Lee, and M. Razaviyayn. Convergence to second-order stationarity for constrained non-convex optimization. *arXiv preprint arXiv:1810.02024*, 2018.

[46] G. Obozinski, L. Jacob, and J.-P. Vert. Group lasso with overlaps: the latent group lasso approach. *arXiv preprint arXiv:1110.0413*, 2011.

[47] N. Parikh, S. Boyd, et al. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

[48] D. Park, A. Kyrillidis, S. Bhojanapalli, C. Caramanis, and S. Sanghavi. Provable burer-monteiro factorization for a class of norm-constrained matrix problems. *arXiv preprint arXiv:1606.01316*, 2016.

[49] G. Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Mathematics of operations research*, 23(2):339–358, 1998.

[50] J. Peng and Y. Wei. Approximating K–means–type clustering via semidefinite programming. *SIAM J. Optim.*, 18(1):186–205, 2007.

[51] M. J. Powell. A method for nonlinear constraints in minimization problems. *Optimization*, pages 283–298, 1969.

[52] A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *ArXiv e-prints*, Nov. 2015.

[53] P. Raghavendra. Optimal algorithms and inapproximability results for every csp? In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 245–254. ACM, 2008.

[54] R. T. Rockafellar. Lagrange multipliers and optimality. *SIAM review*, 35(2):183–238, 1993.

[55] P. Samangouei, M. Kabkab, and R. Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018.

[56] A. Singer. Angular synchronization by eigenvectors and semidefinite programming. *Applied and computational harmonic analysis*, 30(1):20, 2011.

[57] A. Singer and Y. Shkolnisky. Three-dimensional structure determination from common lines in cryo-em by eigenvectors and semidefinite programming. *SIAM journal on imaging sciences*, 4(2):543–572, 2011.

[58] L. Song, A. Smola, A. Gretton, and K. M. Borgwardt. A dependence maximization view of clustering. In *Proceedings of the 24th international conference on Machine learning*, pages 815–822. ACM, 2007.

[59] M. Tepper, A. M. Sengupta, and D. Chklovskii. Clustering is semidefinitely not that hard: Nonnegative sdp for manifold disentangling. *Journal of Machine Learning Research*, 19(82), 2018.

[60] Q. Tran-Dinh, A. Alacaoglu, O. Fercoq, and V. Cevher. An adaptive primal-dual framework for nonsmooth convex minimization. *arXiv preprint arXiv:1808.04648*, 2018.

[61] Q. Tran-Dinh, O. Fercoq, and V. Cevher. A smooth primal-dual optimization framework for nonsmooth composite convex minimization. *SIAM Journal on Optimization*, 28(1):96–134, 2018.

[62] I. Waldspurger and A. Waters. Rank optimality for the burer-monteiro factorization. *arXiv preprint arXiv:1812.03046*, 2018.

[63] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[64] Y. Xu. Iteration complexity of inexact augmented lagrangian methods for constrained convex programming. *arXiv preprint arXiv:1711.05812v2*, 2017.

[65] Y. Xu and W. Yin. A globally convergent algorithm for nonconvex optimization based on block coordinate update. *Journal of Scientific Computing*, 72(2):700–734, 2017.

[66] L. Yang, D. Sun, and K.-C. Toh. Sdpnal+: a majorized semismooth newton-cg augmented lagrangian method for semidefinite programming with nonnegative constraints. *Mathematical Programming Computation*, 7(3):331–366, 2015.

[67] A. Yurtsever, Q. T. Dinh, and V. Cevher. A universal primal-dual convex optimization framework. In *Advances in Neural Information Processing Systems*, pages 3150–3158, 2015.

[68] A. Yurtsever, O. Fercoq, F. Locatello, and V. Cevher. A conditional gradient framework for composite convex minimization with applications to semidefinite programming. *arXiv preprint arXiv:1804.08544*, 2018.

[69] Q. Zhao, S. E. Karisch, F. Rendl, and H. Wolkowicz. Semidefinite programming relaxations for the quadratic assignment problem. *Journal of Combinatorial Optimization*, 2(1):71–109, 1998.

A Complexity Results

A.1 First-Order Optimality

Let us first consider the case where the solver in Step 2 is is the first-order algorithm APGM, described in detail in [27]. At a high level, APGM makes use of $\nabla_x \mathcal{L}_\beta(x,y)$ in (6), the proximal operator prox_g, and the classical Nesterov acceleration [43] to reach first-order stationarity for the subproblem in (7). Suppose that $g = \delta_{\mathcal{X}}$ is the indicator function on a bounded convex set $\mathcal{X} \subset \mathbb{R}^d$ and let

$$\rho = \max_{x\in\mathcal{X}} \|x\|, \tag{26}$$

be the radius of a ball centered at the origin that includes $\mathcal{X}$. Then, adapting the results in [27] to our setup, APGM reaches x_k in Step 2 of Algorithm 1 after

$$\mathcal{O}\left(\frac{\lambda_{\beta_k}^2 \rho^2}{\epsilon_{k+1}}\right) \tag{27}$$

(inner) iterations, where λ_{β_k} denotes the Lipschitz constant of $\nabla_x \mathcal{L}_{\beta_k}(x,y)$, bounded in (15). For the clarity of the presentation, we have used a looser bound in (27) compared to [27]. Using (27), we derive the following corollary, describing the total iteration complexity of Algorithm 1 in terms of the number calls made to the first-order oracle in APGM.

Corollary A.1. *For $b > 1$, let $\beta_k = b^k$ for every k. If we use APGM from [27] for Step 2 of Algorithm 1, the algorithm finds an (ϵ_f, β_k) first-order stationary point, after T calls to the first-order oracle, where*

$$T = \mathcal{O}\left(\frac{Q^3\rho^2}{\epsilon^3}\log_b\left(\frac{Q}{\epsilon}\right)\right) = \tilde{\mathcal{O}}\left(\frac{Q^3\rho^2}{\epsilon^3}\right). \tag{28}$$

Proof. Let K denote the number of (outer) iterations of Algorithm 1 and let ϵ_f denote the desired accuracy of Algorithm 1, see (**??**). Recalling Theorem 4.1, we can then write that

$$\epsilon_f = \frac{Q}{\beta_K}, \tag{29}$$

or, equivalently, $\beta_K = Q/\epsilon_f$. We now count the number of total (inner) iterations T of Algorithm 1 to reach the accuracy ϵ_f. From (15) and for sufficiently large k, recall that $\lambda_{\beta_k} \le \lambda''\beta_k$ is the smoothness parameter of the augmented Lagrangian. Then, from (27) ad by summing over the outer iterations, we bound the total number of (inner) iterations of Algorithm 1 as

$$\begin{aligned}
T &= \sum_{k=1}^{K} \mathcal{O}\left(\frac{\lambda_{\beta_{k-1}}^2\rho^2}{\epsilon_k}\right) \\
&= \sum_{k=1}^{K} \mathcal{O}\left(\beta_{k-1}^3\rho^2\right) \qquad \text{(Step 1 of Algorithm 1)} \\
&\le \mathcal{O}\left(K\beta_{K-1}^3\rho^2\right) \qquad (\{\beta_k\}_k \text{ is increasing}) \\
&\le \mathcal{O}\left(\frac{KQ^3\rho^2}{\epsilon_f^3}\right). \qquad \text{(see (29))}
\end{aligned} \tag{30}$$

In addition, if we specify $\beta_k = b^k$ for all k, we can further refine T. Indeed,

$$\beta_K = b^K \implies K = \log_b\left(\frac{Q}{\epsilon_f}\right), \tag{31}$$

which, after substituting into (30) gives the final bound in Corollary 4.2. □

A.2 Second-Order Optimality

Let us now consider the second-order optimality case where the solver in Step 2 is the the trust region method developed in [17]. Trust region method minimizes a quadratic approximation of the function within a dynamically updated trust-region radius. Second-order trust region method that we consider in this section makes use of Hessian (or an approximation of Hessian) of the augmented Lagrangian in addition to first order oracles.

As shown in [45], finding approximate second-order stationary points of convex-constrained problems is in general NP-hard. For this reason, we focus in this section on the special case of (1) with $g = 0$.

Let us compute the total computational complexity of Algorithm 1 with the trust region method in Step 2, in terms of the number of calls made to the second-order oracle. By adapting the result in [17] to our setup, we find that the number of (inner) iterations required in Step 2 of Algorithm 1 to produce x_{k+1} is

$$\mathcal{O}\left(\frac{\lambda^2_{\beta_k,H}(\mathcal{L}_{\beta_k}(x_1,y) - \min_x \mathcal{L}_{\beta_k}(x,y))}{\epsilon_k^3}\right), \tag{32}$$

where $\lambda_{\beta,H}$ is the Lipschitz constant of the Hessian of the augmented Lagrangian, which is of the order of β, as can be proven similar to Lemma 2.1 and x_1 is the initial iterate of the given outer loop. In [17], the term $\mathcal{L}_\beta(x_1,y) - \min_x \mathcal{L}_\beta(x,y)$ is bounded by a constant independent of ϵ. We assume a uniform bound for this quantity for every β_k, instead of for one value of β_k as in [17]. Using (32) and Theorem 4.1, we arrive at the following:

Corollary A.2. *For $b > 1$, let $\beta_k = b^k$ for every k. We assume that*

$$\mathcal{L}_\beta(x_1,y) - \min_x \mathcal{L}_\beta(x,y) \le L_u, \qquad \forall \beta. \tag{33}$$

If we use the trust region method from [17] for Step 2 of Algorithm 1, the algorithm finds an ϵ-second-order stationary point of (1) in T calls to the second-order oracle where

$$T = \mathcal{O}\left(\frac{L_u Q'^5}{\epsilon^5}\log_b\left(\frac{Q'}{\epsilon}\right)\right) = \widetilde{\mathcal{O}}\left(\frac{L_u Q'^5}{\epsilon^5}\right). \tag{34}$$

Before closing this section, we note that the remark after Corollary 4.2 applies here as well.

A.3 Approximate optimality of (1).

Corollary 4.2 establishes the iteration complexity of Algorithm 1 to reach approximate first-order stationarity for the equivalent formulation of (1) presented in (5). Unlike the exact case, approximate first-order stationarity in (5) does not immediately lend itself to approximate stationarity in (1), and the study of approximate stationarity for the penalized problem (special case of our setting with dual variable set to 0) has also precedent in [5].

However, it is not difficult to verify that, with the more aggressive regime of $\epsilon_{k+1} = 1/\beta_k^2$ in Step 1 of Algorithm 1, one can achieve ϵ-first-order stationarity for (1) with the iteration complexity of $T = \tilde{O}(Q^3\rho^2/\epsilon^6)$ in Corollary 4.2. Note that this conversion is by a naive computation using loose bounds rather than using duality arguments for a tight conversion. For a precedent in convex optimization for relating the convergence in augmented Lagrangian to the constrained problem using duality, see [61].

For the second-order case, it is in general not possible to establish approximate second-order optimality for (5) from Corollary 4.3, with the exception of linear constraints.

B Proof of Theorem 4.1

For every $k \ge 2$, recall from (6) and Step 2 of Algorithm 1 that x_k satisfies

$$\begin{aligned}&\operatorname{dist}(-\nabla f(x_k) - DA(x_k)^\top y_{k-1}\\ &\qquad - \beta_{k-1}DA(x_k)^\top A(x_k), \partial g(x_k))\\ &= \operatorname{dist}(-\nabla_x \mathcal{L}_{\beta_{k-1}}(x_k, y_{k-1}), \partial g(x_k)) \le \epsilon_k.\end{aligned} \tag{35}$$

With an application of the triangle inequality, it follows that

$$\begin{aligned}&\operatorname{dist}(-\beta_{k-1}DA(x_k)^\top A(x_k), \partial g(x_k))\\ &\qquad \le \|\nabla f(x_k)\| + \|DA(x_k)^\top y_{k-1}\| + \epsilon_k,\end{aligned} \tag{36}$$

which in turn implies that

$$\begin{aligned}
&\mathrm{dist}(-DA(x_k)^\top A(x_k), \partial g(x_k)/\beta_{k-1}) \\
&\le \frac{\|\nabla f(x_k)\|}{\beta_{k-1}} + \frac{\|DA(x_k)^\top y_{k-1}\|}{\beta_{k-1}} + \frac{\epsilon_k}{\beta_{k-1}} \\
&\le \frac{\lambda'_f + \lambda'_A \|y_{k-1}\| + \epsilon_k}{\beta_{k-1}},
\end{aligned} \tag{37}$$

where λ'_f, λ'_A were defined in (17). We next translate (37) into a bound on the feasibility gap $\|A(x_k)\|$. Using the regularity condition (18), the left-hand side of (37) can be bounded below as

$$\mathrm{dist}(-DA(x_k)^\top A(x_k), \partial g(x_k)/\beta_{k-1}) \ge \nu \|A(x_k)\|. \qquad \text{(see (18))} \tag{38}$$

By substituting (38) back into (37), we find that

$$\|A(x_k)\| \le \frac{\lambda'_f + \lambda'_A \|y_{k-1}\| + \epsilon_k}{\nu \beta_{k-1}}. \tag{39}$$

In words, the feasibility gap is directly controlled by the dual sequence $\{y_k\}_k$. We next establish that the dual sequence is bounded. Indeed, for every $k \in K$, note that

$$\begin{aligned}
\|y_k\| &= \|y_0 + \sum_{i=1}^{k} \sigma_i A(x_i)\| \quad \text{(Step 5 of Algorithm 1)} \\
&\le \|y_0\| + \sum_{i=1}^{k} \sigma_i \|A(x_i)\| \qquad \text{(triangle inequality)} \\
&\le \|y_0\| + \sum_{i=1}^{k} \frac{\|A(x_1)\| \log^2 2}{k \log^2(k+1)} \quad \text{(Step 4)} \\
&\le \|y_0\| + c\|A(x_1)\| \log^2 2 =: y_{\max},
\end{aligned} \tag{40}$$

where

$$c \ge \sum_{i=1}^{\infty} \frac{1}{k \log^2(k+1)}. \tag{41}$$

Substituting (40) back into (39), we reach

$$\begin{aligned}
\|A(x_k)\| &\le \frac{\lambda'_f + \lambda'_A y_{\max} + \epsilon_k}{\nu \beta_{k-1}} \\
&\le \frac{2\lambda'_f + 2\lambda'_A y_{\max}}{\nu \beta_{k-1}},
\end{aligned} \tag{42}$$

where the second line above holds if k_0 is large enough, which would in turn guarantees that $\epsilon_k = 1/\beta_{k-1}$ is sufficiently small since $\{\beta_k\}_k$ is increasing and unbounded. It remains to control the first term in (**??**). To that end, after recalling Step 2 of Algorithm 1 and applying the triangle inequality, we can write that

$$\begin{aligned}
&\mathrm{dist}(-\nabla_x \mathcal{L}_{\beta_{k-1}}(x_k, y_k), \partial g(x_k)) \\
&\le \mathrm{dist}(-\nabla_x \mathcal{L}_{\beta_{k-1}}(x_k, y_{k-1}), \partial g(x_k)) \\
&+ \|\nabla_x \mathcal{L}_{\beta_{k-1}}(x_k, y_k) - \nabla_x \mathcal{L}_{\beta_{k-1}}(x_k, y_{k-1})\|.
\end{aligned} \tag{43}$$

The first term on the right-hand side above is bounded by ϵ_k, by Step 5 of Algorithm 1. For the second term on the right-hand side of (43), we write that

$$\begin{aligned}
&\|\nabla_x \mathcal{L}_{\beta_{k-1}}(x_k, y_k) - \nabla_x \mathcal{L}_{\beta_{k-1}}(x_k, y_{k-1})\| \\
&= \|DA(x_k)^\top (y_k - y_{k-1})\| \qquad \text{(see (6))} \\
&\le \lambda'_A \|y_k - y_{k-1}\| \qquad \text{(see (17))} \\
&= \lambda'_A \sigma_k \|A(x_k)\| \qquad \text{(see Step 5 of Algorithm 1)} \\
&\le \frac{2\lambda'_A \sigma_k}{\nu \beta_{k-1}} (\lambda'_f + \lambda'_A y_{\max}). \qquad \text{(see (42))}
\end{aligned} \tag{44}$$

By combining (43,44), we find that

$$\begin{aligned}&\operatorname{dist}(\nabla_x \mathcal{L}_{\beta_{k-1}}(x_k, y_k), \partial g(x_k))\\&\le \frac{2\lambda'_A \sigma_k}{\nu \beta_{k-1}} (\lambda'_f + \lambda'_A y_{\max}) + \epsilon_k.\end{aligned} \tag{45}$$

By combining (42,45), we find that

$$\begin{aligned}&\operatorname{dist}(-\nabla_x \mathcal{L}_{\beta_{k-1}}(x_k, y_k), \partial g(x_k)) + \|A(x_k)\|\\&\le \left(\frac{2\lambda'_A \sigma_k}{\nu \beta_{k-1}} (\lambda'_f + \lambda'_A y_{\max}) + \epsilon_k\right)\\&\quad + 2\left(\frac{\lambda'_f + \lambda'_A y_{\max}}{\nu \beta_{k-1}}\right).\end{aligned} \tag{46}$$

Applying $\sigma_k \le \sigma_1$, we find that

$$\begin{aligned}&\operatorname{dist}(-\nabla_x \mathcal{L}_{\beta_{k-1}}(x_k, y_k), \partial g(x_k)) + \|A(x_k)\|\\&\le \frac{2\lambda'_A \sigma_1 + 2}{\nu \beta_{k-1}} (\lambda'_f + \lambda'_A y_{\max}) + \epsilon_k.\end{aligned} \tag{47}$$

For the second part of the theorem, we use the Weyl's inequality and Step 5 of Algorithm 1 to write

$$\begin{aligned}\lambda_{\min}(\nabla_{xx} \mathcal{L}_{\beta_{k-1}}(x_k, y_{k-1})) &\ge \lambda_{\min}(\nabla_{xx} \mathcal{L}_{\beta_{k-1}}(x_k, y_k))\\&- \sigma_k \|\sum_{i=1}^m A_i(x_k) \nabla^2 A_i(x_k)\|.\end{aligned} \tag{48}$$

The first term on the right-hand side is lower bounded by $-\epsilon_{k-1}$ by Step 2 of Algorithm 1. We next bound the second term on the right-hand side above as

$$\begin{aligned}&\sigma_k \|\sum_{i=1}^m A_i(x_k) \nabla^2 A_i(x_k)\|\\&\le \sigma_k \sqrt{m} \max_i \|A_i(x_k)\| \|\nabla^2 A_i(x_k)\|\\&\le \sigma_k \sqrt{m} \lambda_A \frac{2\lambda'_f + 2\lambda'_A y_{\max}}{\nu \beta_{k-1}},\end{aligned}$$

where the last inequality is due to (4,42). Plugging into (48) gives

$$\begin{aligned}&\lambda_{\min}(\nabla_{xx} \mathcal{L}_{\beta_{k-1}}(x_k, y_{k-1}))\\&\ge -\epsilon_{k-1} - \sigma_k \sqrt{m} \lambda_A \frac{2\lambda'_f + 2\lambda'_A y_{\max}}{\nu \beta_{k-1}},\end{aligned}$$

which completes the proof of Theorem 4.1.

C Proof of Lemma 2.1

Proof. Note that

$$\mathcal{L}_\beta(x, y) = f(x) + \sum_{i=1}^m y_i A_i(x) + \frac{\beta}{2} \sum_{i=1}^m (A_i(x))^2, \tag{49}$$

which implies that

$$\begin{aligned}&\nabla_x \mathcal{L}_\beta(x, y)\\&= \nabla f(x) + \sum_{i=1}^m y_i \nabla A_i(x) + \frac{\beta}{2} \sum_{i=1}^m A_i(x) \nabla A_i(x)\\&= \nabla f(x) + DA(x)^\top y + \beta DA(x)^\top A(x),\end{aligned} \tag{50}$$

where $DA(x)$ is the Jacobian of A at x. By taking another derivative with respect to x, we reach

$$\nabla_x^2 \mathcal{L}_\beta(x,y) = \nabla^2 f(x) + \sum_{i=1}^m (y_i + \beta A_i(x)) \nabla^2 A_i(x) \\ + \beta \sum_{i=1}^m \nabla A_i(x) \nabla A_i(x)^\top. \tag{51}$$

It follows that

$$\begin{aligned} &\|\nabla_x^2 \mathcal{L}_\beta(x,y)\| \\ &\le \|\nabla^2 f(x)\| + \max_i \|\nabla^2 A_i(x)\| \left(\|y\|_1 + \beta \|A(x)\|_1\right) \\ &\quad + \beta \sum_{i=1}^m \|\nabla A_i(x)\|^2 \\ &\le \lambda_h + \sqrt{m}\lambda_A \left(\|y\| + \beta\|A(x)\|\right) + \beta \|DA(x)\|_F^2. \end{aligned} \tag{52}$$

For every x such that $\|x\| \le \rho$ and $\|A(x)\| \le \rho$, we conclude that

$$\|\nabla_x^2 \mathcal{L}_\beta(x,y)\| \le \lambda_f + \sqrt{m}\lambda_A \left(\|y\| + \beta\rho'\right) + \beta \max_{\|x\|\le\rho} \|DA(x)\|_F^2, \tag{53}$$

which completes the proof of Lemma 2.1. □

D Clustering

We only verify the condition in (18) here. Note that

$$A(x) = VV^\top \mathbf{1} - \mathbf{1}, \tag{54}$$

$$\begin{aligned} DA(x) &= \begin{bmatrix} w_{1,1}x_1^\top & \cdots & w_{1,n}x_1^\top \\ \vdots & & \\ w_{n,1}x_n^\top & \cdots & w_{n,n}1x_n^\top \end{bmatrix} \\ &= \begin{bmatrix} V & \cdots & V \end{bmatrix} + \begin{bmatrix} x_1^\top & & \\ & \ddots & \\ & & x_n^\top \end{bmatrix}, \end{aligned} \tag{55}$$

where $w_{i.i} = 2$ and $w_{i,j} = 1$ for $i \neq j$. In the last line above, n copies of V appear and the last matrix above is block-diagonal. For x_k, define V_k accordingly and let $x_{k,i}$ be the ith row of V_k. Consequently,

$$\begin{aligned} DA(x_k)^\top A(x_k) &= \begin{bmatrix} (V_k^\top V_k - I_n)V_k^\top \mathbf{1} \\ \vdots \\ (V_k^\top V_k - I_n)V_k^\top \mathbf{1} \end{bmatrix} \\ &\quad + \begin{bmatrix} x_{k,1}(V_k V_k^\top \mathbf{1} - \mathbf{1})_1 \\ \vdots \\ x_{k,n}(V_k V_k^\top \mathbf{1} - \mathbf{1})_n \end{bmatrix}, \end{aligned} \tag{56}$$

where $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix. Let us make a number of simplifying assumptions. First, we assume that $\|x_k\| < \sqrt{s}$ (which can be enforced in the iterates by replacing C with $(1-\epsilon)C$ for a small positive ϵ in the subproblems). Under this assumption, it follows that

$$(\partial g(x_k))_i = \begin{cases} 0 & (x_k)_i > 0 \\ \{a : a \le 0\} & (x_k)_i = 0, \end{cases} \qquad i \le d. \tag{57}$$

Second, we assume that V_k has nearly orthonormal columns, namely, $V_k^\top V_k \approx I_n$. This can also be enforced in each iterate of Algorithm 1 and naturally corresponds to well-separated clusters. While a more fine-tuned argument can remove these assumptions, they will help us simplify the presentation here. Under these assumptions, the (squared) right-hand side of (18) becomes

$$
\begin{aligned}
&\operatorname{dist}\left(-DA(x_k)^\top A(x_k), \frac{\partial g(x_k)}{\beta_{k-1}}\right)^2 \\
&= \left\| \left(-DA(x_k)^\top A(x_k)\right)_+ \right\|^2 \qquad (a_+ = \max(a,0)) \\
&= \left\| \begin{bmatrix} x_{k,1}(V_k V_k^\top \mathbf{1} - \mathbf{1})_1 \\ \vdots \\ x_{k,n}(V_k V_k^\top \mathbf{1} - \mathbf{1})_n \end{bmatrix} \right\|^2 \qquad (x_k \in C \Rightarrow x_k \geq 0) \\
&= \sum_{i=1}^n \|x_{k,i}\|^2 (V_k V_k^\top \mathbf{1} - \mathbf{1})_i^2 \\
&\geq \min_i \|x_{k,i}\|^2 \cdot \sum_{i=1}^n (V_k V_k^\top \mathbf{1} - \mathbf{1})_i^2 \\
&= \min_i \|x_{k,i}\|^2 \cdot \|V_k V_k^\top \mathbf{1} - \mathbf{1}\|^2. \qquad (58)
\end{aligned}
$$

Therefore, given a prescribed ν, ensuring $\min_i \|x_{k,i}\| \geq \nu$ guarantees (18). When the algorithm is initialized close enough to the constraint set, there is indeed no need to separately enforce (58). In practice, often n exceeds the number of true clusters and a more intricate analysis is required to establish (18) by restricting the argument to a particular subspace of $\mathbb{R}^n$.

E Additional Experiments

E.1 Basis Pursuit

Basis Pursuit (BP) finds sparsest solutions of an under-determined system of linear equations by solving

$$\min_z \|z\|_1 \quad \text{s.t.} \quad Bz = b, \tag{59}$$

where $B \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$. Various primal-dual convex optimization algorithms are available in the literature to solve BP, including [60, 19]. We compare our algorithm against state-of-the-art primal-dual convex methods for solving (59), namely, Chambole-Pock [19], ASGARD [61] and ASGARD-DL [60].

Here, we take a different approach and cast (59) as an instance of (1). Note that any $z \in \mathbb{R}^d$ can be decomposed as $z = z^+ - z^-$, where $z^+, z^- \in \mathbb{R}^d$ are the positive and negative parts of z, respectively. Then consider the change of variables $z^+ = u_1^{\circ 2}$ and $z^- = u_2^{\circ 2} \in \mathbb{R}^d$, where $\circ$ denotes element-wise power. Next, we concatenate u_1 and u_2 as $x := [u_1^\top, u_2^\top]^\top \in \mathbb{R}^{2d}$ and define $\overline{B} := [B, -B] \in \mathbb{R}^{n\times 2d}$. Then, (59) is equivalent to (1) with

$$f(x) = \|x\|^2, \quad g(x) = 0, \quad \text{s.t.} \quad A(x) = \overline{B}x^{\circ 2} - b. \tag{60}$$

We draw the entries of B independently from a zero-mean and unit-variance Gaussian distribution. For a fixed sparsity level k, the support of $z_* \in \mathbb{R}^d$ and its nonzero amplitudes are also drawn from the standard Gaussian distribution. Then the measurement vector is created as $b = Bz + \epsilon$, where ϵ is the noise vector with entries drawn independently from the zero-mean Gaussian distribution with variance $\sigma^2 = 10^{-6}$.

The results are compiled in Figure 2. Clearly, the performance of Algorithm 1 with a second-order solver for BP is comparable to the rest. It is, indeed, interesting to see that these type of nonconvex relaxations gives the solution of convex one and first order methods succeed.

Discussion: The true potential of our reformulation is in dealing with more structured norms rather than ℓ_1, where computing the proximal operator is often intractable. One such case is the latent group

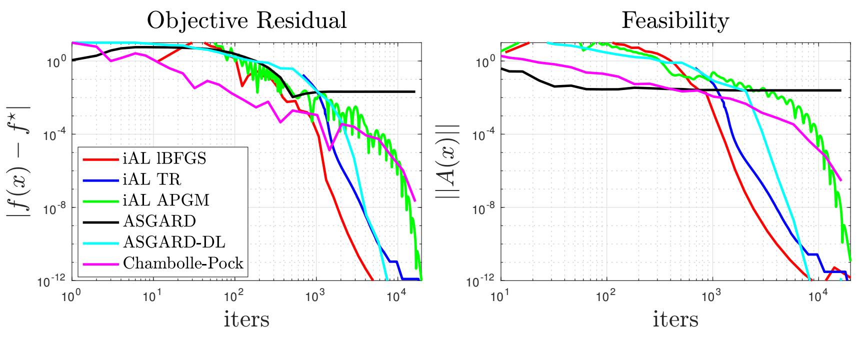

Figure 2: Basis Pursuit

lasso norm [46], defined as

$$\|z\|_\Omega = \sum_{i=1}^{I} \|z_{\Omega_i}\|,$$

where $\{\Omega_i\}_{i=1}^I$ are (not necessarily disjoint) index sets of $\{1, \cdots, d\}$. Although not studied here, we believe that the nonconvex framework presented in this paper can serve to solve more complicated problems, such as the latent group lasso. We leave this research direction for future work.

Condition verification: In the sequel, we verify that Theorem 4.1 indeed applies to (1) with the above f, A, g. Note that

$$DA(x) = 2\overline{B}\text{diag}(x), \tag{61}$$

where $\text{diag}(x) \in \mathbb{R}^{2d\times 2d}$ is the diagonal matrix formed by x. The left-hand side of (18) then reads as

$$\begin{aligned}
&\text{dist}\left(-DA(x_k)^\top A(x_k), \frac{\partial g(x_k)}{\beta_{k-1}}\right) \\
&= \text{dist}\left(-DA(x_k)^\top A(x_k), \{0\}\right) \qquad (g \equiv 0) \\
&= \|DA(x_k)^\top A(x_k)\| \\
&= 2\|\text{diag}(x_k)\overline{B}^\top(\overline{B}x_k^{\circ 2} - b)\|. \qquad (\text{see } (61))
\end{aligned} \tag{62}$$

To bound the last line above, let x_* be a solution of (1) and note that $\overline{B}x_*^{\circ 2} = b$ by definition. Let also $z_k, z_* \in \mathbb{R}^d$ denote the vectors corresponding to x_k, x_*. Corresponding to x_k, also define $u_{k,1}, u_{k,2}$ naturally and let $|z_k| = u_{k,1}^{\circ 2} + u_{k,2}^{\circ 2} \in \mathbb{R}^d$ be the vector of amplitudes of z_k. To simplify matters, let us assume also that B is full-rank. We then rewrite the norm in the last line of (62) as

$$\begin{aligned}
&\|\text{diag}(x_k)\overline{B}^\top(\overline{B}x_k^{\circ 2} - b)\|^2 \\
&= \|\text{diag}(x_k)\overline{B}^\top\overline{B}(x_k^{\circ 2} - x_*^{\circ 2})\|^2 \qquad (\overline{B}x_*^{\circ 2} = b) \\
&= \|\text{diag}(x_k)\overline{B}^\top B(x_k - x_*)\|^2 \\
&= \|\text{diag}(u_{k,1})B^\top B(z_k - z_*)\|^2 \\
&\qquad + \|\text{diag}(u_{k,2})B^\top B(z_k - z_*)\|^2 \\
&= \|\text{diag}(u_{k,1}^{\circ 2} + u_{k,2}^{\circ 2})B^\top B(z_k - z_*)\|^2 \\
&= \|\text{diag}(|z_k|)B^\top B(z_k - z_*)\|^2 \\
&\geq \eta_n(B\text{diag}(|z_k|))^2\|B(z_k - z_*)\|^2 \\
&= \eta_n(B\text{diag}(|z_k|))^2\|Bz_k - b\|^2 \qquad (Bz_* = \overline{B}x_*^{\circ 2} = b) \\
&\geq \min_{|T|=n} \eta_n(B_T) \cdot |z_{k,(n)}|^2\|Bz_k - b\|^2,
\end{aligned} \tag{63}$$

where $\eta_n(\cdot)$ returns the nth largest singular value of its argument. In the last line above, B_T is the restriction of B to the columns indexed by T of size n. Moreover, $z_{k,(n)}$ is the nth largest entry of z in magnitude. Given a prescribed ν, (18) therefore holds if

$$|z_{k,(n)}| \geq \frac{\nu}{2\sqrt{\min_{|T|=n} \eta_n(B_T)}}, \tag{64}$$

for every iteration k. If Algorithm 1 is initialized close enough to the solution z^* and the entries of z^* are sufficiently large in magnitude, there will be no need to directly enforce (64).

E.2 Generalized Eigenvalue Problem

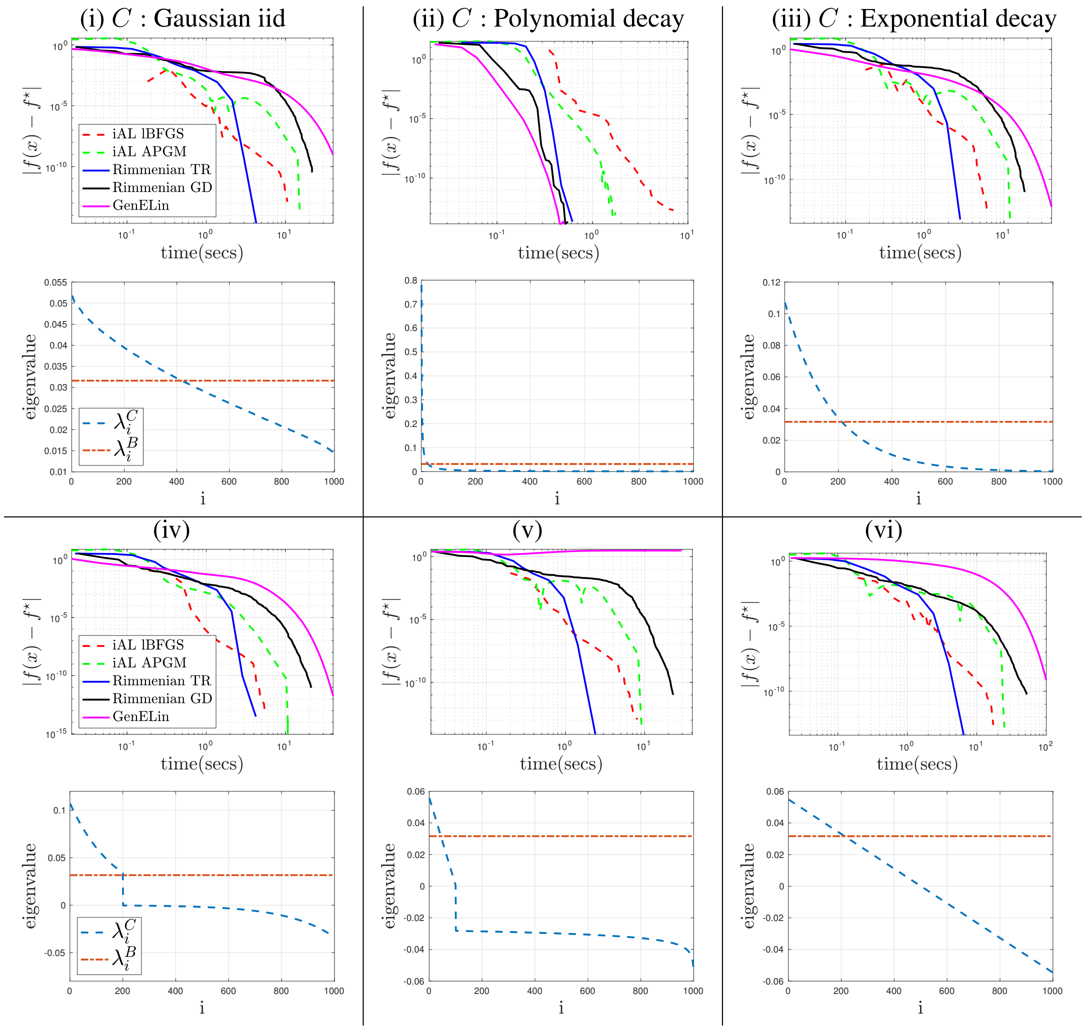

Figure 3: *(Top)* Objective convergence for calculating top generalized eigenvalue and eigenvector of B and C. *(Bottom)* Eigenvalue structure of the matrices. For (i),(ii) and (iii), C is positive semidefinite; for (iv), (v) and (vi), C contains negative eigenvalues. [(i): Generated by taking symmetric part of iid Gaussian matrix. (ii): Generated by randomly rotating diag$(1^{-p}, 2^{-p}, \cdots, 1000^{-p})(p = 1)$. (iii): Generated by randomly rotating diag$(10^{-p}, 10^{-2p}, \cdots, 10^{-1000p})(p = 0.0025)$.]

Generalized eigenvalue problem has extensive applications in machine learning, statistics and data analysis [26]. The well-known nonconvex formulation of the problem is [13] given by

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} x^\top C x \\ x^\top B x = 1, \end{cases} \tag{65}$$

where $B, C \in \mathbb{R}^{n\times n}$ are symmetric matrices and B is positive definite, namely, $B \succ 0$. The generalized eigenvector computation is equivalent to performing principal component analysis (PCA) of C in the norm B. It is also equivalent to computing the top eigenvector of symmetric matrix $S = B^{-1/2}CB^{1/2}$ and multiplying the resulting vector by $B^{-1/2}$. However, for large values of n, computing $B^{-1/2}$ is extremely expensive. The natural convex SDP relaxation for (65) involves lifting $Y = xx^\top$ and removing the nonconvex $\text{rank}(Y) = 1$ constraint, namely,

$$\begin{cases} \min\limits_{Y\in\mathbb{R}^{n\times n}} \text{tr}(CY) \\ \text{tr}(BY) = 1, \quad X \succeq 0. \end{cases} \tag{66}$$

Here, however, we opt to directly solve (65) because it fits into our template with

$$\begin{aligned} f(x) =& x^\top Cx, \quad g(x) = 0, \\ A(x) =& x^\top Bx - 1. \end{aligned} \tag{67}$$

We compare our approach against three different methods: manifold based Riemannian gradient descent and Riemannian trust region methods in [11] and the linear system solver in [26], abbrevated as GenELin. We have used Manopt software package in [**?**] for the manifold based methods. For GenELin, we have utilized Matlab's backslash operator as the linear solver. The results are compiled in Figure 3.

Condition verification: Here, we verify the regularity condition in (18) for problem (65). Note that

$$DA(x) = (2Bx)^\top. \tag{68}$$

Therefore,

$$\begin{aligned} \text{dist}\left(-DA(x_k)^\top A(x_k), \frac{\partial g(x_k)}{\beta_{k-1}}\right)^2 &= \text{dist}\left(-DA(x_k)^\top A(x_k), \{0\}\right)^2 \qquad (g \equiv 0) \\ &= \|DA(x_k)^\top A(x_k)\|^2 \\ &= \|2Bx_k(x_k^\top Bx_k - 1)\|^2 \qquad \text{(see (68))} \\ &= 4(x_k^\top Bx_k - 1)^2\|Bx_k\|^2 \\ &= 4\|Bx_k\|^2\|A(x_k)\|^2 \qquad \text{(see (67))} \\ &\geq \eta_{\min}(B)^2\|x_k\|^2\|A(x_k)\|^2, \end{aligned} \tag{69}$$

where $\eta_{\min}(B)$ is the smallest eigenvalue of the positive definite matrix B. Therefore, for a prescribed ν, the regularity condition in (18) holds with $\|x_k\| \geq \nu/\eta_{min}$ for every k. If the algorithm is initialized close enough to the constraint set, there will be again no need to directly enforce this latter condition.

E.3 ℓ_∞ Denoising with a Generative Prior

The authors of [55, 30] have proposed to project onto the range of a Generative Adversarial network (GAN) [28], as a way to defend against adversarial examples. For a given noisy observation $x^* + \eta$, they consider a projection in the ℓ_2 norm. We instead propose to use our augmented Lagrangian method to denoise in the ℓ_∞ norm, a much harder task:

$$\begin{aligned} \min_{x,z} \quad & \|x^* + \eta - x\|_\infty \\ \text{s.t.} \quad & x = G(z). \end{aligned} \tag{70}$$

We use a pretrained generator for the MNIST dataset, given by a standard deconvolutional neural network architecture [52]. We compare the succesful optimizer Adam [34] and gradient Descent against our method. Our algorithm involves two forward and one backward pass through the network, as oposed to Adam that requires only one forward/backward pass. For this reason we let our algorithm run for 2000 iterations, and Adam and GD for 3000 iterations. Both Adam and gradient descent generate a sequence of feasible iterates $x_t = G(z_t)$. For this reason we plot the objective evaluated at the point $G(z_t)$ vs iteration count in figure 4. Our method successfully minimizes the objective value, while Adam and GD do not.

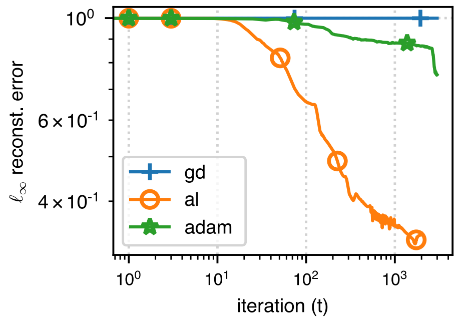

Figure 4: Augmented Lagrangian vs Adam and Gradient descent for ℓ_∞ denoising

E.4 Quadratic assginment problem

Let K, L be $n \times n$ symmetric metrices. QAP in its simplest form can be written as

$$\max \operatorname{tr}(KPLP), \quad \text{subject to } P \text{ be a permutation matrix} \tag{71}$$

A direct approach for solving (71) involves a combinatorial search. To get the SDP relaxation of (71), we will first lift the QAP to a problem involving a larger matrix. Observe that the objective function takes the form

$$\operatorname{tr}((K \otimes L)(\operatorname{vec}(P)\operatorname{vec}(P^\top))),$$

where $\otimes$ denotes the Kronecker product. Therefore, we can recast (71) as

$$\operatorname{tr}((K \otimes L)Y) \quad \text{subject to } \; Y = \operatorname{vec}(P)\operatorname{vec}(P^\top), \tag{72}$$

where P is a permutation matrix. We can relax the equality constraint in (72) to a semidefinite constraint and write it in an equivalent form as

$$X = \begin{bmatrix} 1 & \operatorname{vec}(P)^\top \\ \operatorname{vec}(P) & Y \end{bmatrix} \succeq 0 \;\; \text{for a symmetric} X \in \mathbb{S}^{(n^2+1)\times(n^2+1)}$$

We now introduce the following constraints such that

$$B_k(X) = \mathbf{b_k}, \quad \mathbf{b_k} \in \mathbb{R}^{m_k} \tag{73}$$

to make sure X has a proper structure. Here, B_k is a linear operator on X and the total number of constraints is $m = \sum_k m_k$. Hence, SDP relaxation of the quadratic assignment problem takes the form,

$$\begin{aligned} \max \;\; & \langle C, X \rangle \\ \text{subject to } \;\; & P1 = 1, \;\; 1^\top P = 1, \;\; P \geq 0 \\ & \operatorname{trace}_1(Y) = I \;\; \operatorname{trace}_2(Y) = I \\ & \operatorname{vec}(P) = \operatorname{diag}(Y) \\ & \operatorname{trace}(Y) = n \;\; \begin{bmatrix} 1 & \operatorname{vec}(P)^\top \\ \operatorname{vec}(P) & Y \end{bmatrix} \succeq 0, \end{aligned} \tag{74}$$

where $\operatorname{trace}_1(.)$ and $\operatorname{trace}_2(.)$ are partial traces satisfying,

$$\operatorname{trace}_1(K \otimes L) = \operatorname{trace}(K)L \quad \text{and} \quad \operatorname{trace}_2(K \otimes L) = K\operatorname{trace}(L)$$

$$\text{trace}_1^*(T) = I \otimes T \quad \text{and} \quad \text{trace}_2^*(T) = T \otimes I$$

$1st$ set of equalities are due to the fact that permutation matrices are doubly stochastic. $2nd$ set of equalities are to ensure permutation matrices are orthogonal, i.e., $PP^\top = P^\top P = I$. $3rd$ set of equalities are to enforce every individual entry of the permutation matrix takes either 0 or 1, i.e., $X_{1,i} = X_{i,i} \;\; \forall i \in [1, n^2+1]$. . Trace constraint in the last line is to bound the problem domain. By concatenating the B_k's in (73), we can rewrite (74) in standard SDP form as

$$\begin{aligned} \max \quad & \langle C, X \rangle \\ \text{subject to} \quad & B(X) = \mathbf{b}, \;\; \mathbf{b} \in \mathbb{R}^m \\ & \text{trace}(X) = n+1 \\ & X_{ij} \geq 0, \quad i,j \;\; \mathcal{G} \\ & X \succeq 0, \end{aligned} \tag{75}$$

where $\mathcal{G}$ represents the index set for which we introduce the nonnegativities. When $\mathcal{G}$ covers the wholes set of indices, we get the best approximation to the original problem. However, it becomes computationally undesirable as the problem dimension increases. Hence, we remove the redundant nonnegativity constraints and enforce it for the indices where Kronecker product between K and L is nonzero.

We penalize the non-negativity constraints and add it to the augmented Lagrangian objective since a projection to the positive orthant approach in the low rank space as we did for the clustering does not work here.

We take [23] as the baseline. This is an SDP based approach for solving QAP problems containing a sparse graph. We compare against the best feasible upper bounds reported in [23] for the given instances. Here, optimality gap is defined as

$$\%\text{Gap} = \frac{|\text{bound} - \text{optimal}|}{\text{optimal}} \times 100$$

We used a (relatively) sparse graph data set from the QAP library. We run our low rank algorithm for different rank values. r_m in each instance corresponds to the smallest integer satisfying the Pataki bound [49, 1]. Results are shown in Table 1. Primal feasibility values except for the last instance $esc128$ is less than 10^{-5} and we obtained bounds at least as good as the ones reported in [23] for these problems.

For $esc128$, the primal feasibility is $\approx 10^{-1}$, hence, we could not manage to obtain a good optimality gap due to limited time.

		Optimality Gap (%)					
Data	Optimal Value	Sparse QAP [23]	iAL				
			$r = 10$	$r = 25$	$r = 50$	$r = r_m$	r_m
esc16a	68	8.8	11.8	**0**	**0**	5.9	157
esc16b	292	**0**	**0**	**0**	**0**	**0**	224
esc16c	160	5	5.0	5.0	**2.5**	3.8	177
esc16d	16	12.5	37.5	**0**	**0**	25.0	126
esc16e	28	7.1	7.1	**0**	14.3	7.1	126
esc16g	26	**0**	23.1	7.7	**0**	**0**	126
esc16h	996	**0**	**0**	**0**	**0**	**0**	224
esc16i	14	**0**	**0**	**0**	14.3	**0**	113
esc16j	8	**0**	**0**	**0**	**0**	**0**	106
esc32a	130	93.8	129.2	109.2	104.6	**83.1**	433
esc32b	168	88.1	111.9	92.9	97.6	**69.0**	508
esc32c	642	7.8	15.6	14.0	15.0	**4.0**	552
esc32d	200	21	28.0	28.0	29.0	**17.0**	470
esc32e	2	**0**	**0**	**0**	**0**	**0**	220
esc32g	6	**0**	33.3	**0**	**0**	**0**	234
esc32h	438	18.3	25.1	19.6	25.1	**13.2**	570
esc64a	116	53.4	62.1	51.7	58.6	**34.5**	899
esc128	64	**175**	256.3	193.8	243.8	215.6	2045

Table 1: Comparison between upper bounds on the problems from the QAP library with (relatively) sparse L.