ℓ_p-Box ADMM: A Versatile Framework for Integer Programming

Baoyuan Wu, *Member, IEEE*, and Bernard Ghanem, *Member, IEEE*

Abstract—This paper revisits the integer programming (IP) problem, which plays a fundamental role in many computer vision and machine learning applications. The literature abounds with many seminal works that address this problem, some focusing on continuous approaches (e.g., linear program relaxation), while others on discrete ones (e.g., min-cut). However, since many of these methods are designed to solve specific IP forms, they cannot adequately satisfy the simultaneous requirements of accuracy, feasibility, and scalability. To this end, we propose a novel and versatile framework called ℓ_p-box ADMM, which is based on two main ideas. (1) The discrete constraint is equivalently replaced by the intersection of a box and an ℓ_p-norm sphere. (2) We infuse this equivalence into the Alternating Direction Method of Multipliers (ADMM) framework to handle the continuous constraints separately and to harness its attractive properties. More importantly, the ADMM update steps can lead to manageable sub-problems in the continuous domain. To demonstrate its efficacy, we apply it to an optimization form that occurs often in computer vision and machine learning, namely binary quadratic programming (BQP). In this case, the ADMM steps are simple, computationally efficient. Moreover, we present the theoretic analysis about the global convergence of the ℓ_p-box ADMM through adding a perturbation with the sufficiently small factor ϵ to the original IP problem. Specifically, the globally converged solution generated by ℓ_p-box ADMM for the perturbed IP problem will be close to the stationary and feasible point of the original IP problem within $O(\epsilon)$. We demonstrate the applicability of ℓ_p-box ADMM on three important applications: MRF energy minimization, graph matching, and clustering. Results clearly show that it significantly outperforms existing generic IP solvers both in runtime and objective. It also achieves very competitive performance to state-of-the-art methods designed specifically for these applications.

Index Terms— Integer programming, nonconvex optimization, ADMM, computer vision, machine learning

1 Introduction

In this work, we focus on the problem of integer programming (IP), which can be generally formulated as a binary optimization as follows:

$$\min_{\mathbf{x}\in\{0,1\}^n} \; f(\mathbf{x}), \; \text{s.t.} \; \mathbf{x} \in \mathcal{C}. \tag{1}$$

Note that the discrete constraint space could include multiple states (more than two). But, by introducing auxiliary variables and linear equality constraints, it can be easily transformed into the binary constraint space $\{0,1\}^n$ [1]. Therefore, in the rest of the paper, we will consider IP problems that have already been transformed into the binary form in Eq. (1). The additional constraint space $\mathcal{C}$ is application-specific, e.g., in many cases, it is a polyhedron (the intersection of linear equality and inequality constraints).

IP problems abound in the field of computer vision (CV) and machine learning (ML). In many applications, solving a particular form of Eq. (1) is viewed as a fundamental module that researchers use as a plug-and-play routine. A few typical examples include (but not limited to) clustering [2], feature selection [3], image co-segmentation [4], [5], image denoising [6], binary hashing [7], graph matching [8], [9], etc. One popular manifestation of Eq. (1) is the energy minimization of the pairwise MRF model [10], where $f(\mathbf{x})$ is a quadratic function (convex in the continuous domain) and $\mathcal{C}$ enforces that each node takes on only one state. This form alone has been popularized in many labeling problems in CV including stereo matching [11] and automatic and interactive image segmentation [12], [13], [14].

Although many popular tasks in CV and ML fall under the general form of Eq. (1) and the IP literature is rich and ever-evolving, there does not seem to be a general-purpose framework for solving these types of problems, which is scalable and efficient. The situation is quite different in the continuous domain, where many generic and popular optimization methods exist (e.g., interior point methods for smooth problems). Indeed, there do exist efficient and, in some cases, global discrete solutions (e.g., binary MRF energy minimization with submodular weights or unimodular binary linear programs) for some IP forms; however, they only apply to limited types of this general problem. This is probably due to the fact that the problem in its general form is NP-hard. An intuitive approach is to relax the binary constraints to continuous ones, which *approximates* the IP problem with a continuous one. This strategy has the advantage of exploiting well-studied concepts in continuous optimization; however, the drawback is either in high computational complexity or the undesired side effects of thresholding the final continuous solution, or both. Moreover, there is a large body of work that utilizes

- *B. Wu is with the Visual Computing Center, King Abdullah University of Science and Technology, Thuwal 23955-6900, Kingdom of Saudi Arabia, and also with the Tencent AI Lab, Shenzhen 518057, China. E-mail: wubaoyuan1987@gmail.com.*
- *B. Ghanem is with the Visual Computing Center, King Abdullah University of Science and Technology, Thuwal 23955-6900, Kingdom of Saudi Arabia. E-mail: bernard.ghanem@kaust.edu.sa.*

Manuscript received 27 June 2016; revised 11 Feb. 2018; accepted 5 June 2018. Date of publication 10 June 2018; date of current version 12 June 2019. (Corresponding author: Baoyuan Wu.) Recommended for acceptance by S. Vishwanathan. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TPAMI.2018.2845842

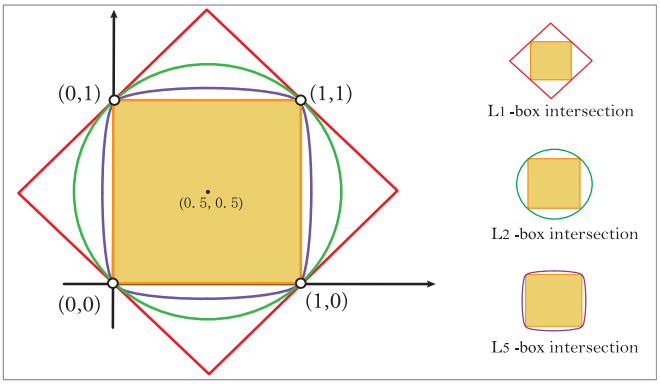

Fig. 1. Geometric illustration of the equivalence between ℓ_p-*box intersection* and the set of binary points in $\mathbb{R}^2$. For clarity, we just show the cases when $p \in \{1, 2, 5\}$.

similar methods to exactly solve the IP problem, such as Branch-and-Bound (BB) [15], cutting plane (CP) [16], and exact penalty methods [17], [18], [19]). Unfortunately, these methods are usually plagued with high computational complexity and/or the risk of getting stuck in undesirable local minima, thus, precluding their use in many practical, medium-to-large scale problems in ML and CV. Therefore, there seems to be an inherent need for a unified framework or tool that researchers can use to reach a desirable (not necessarily global), feasible, and binary solution without sacrificing much computational efficiency. This work can be considered as an insightful and practical step in that direction.

In this paper, we propose to handle the binary constraints in Eq. (1) by replacing them with an equivalent set of continuous constraints, namely the intersection between the box (n-convex constraints) and the shifted ℓ_p-sphere (a non-convex constraint), of which a geometric interpretation is shown in Fig. 1.

Proposition 1. *ℓ_p-Box Intersection: The binary set $\{0,1\}^n$ can be equivalently replaced by the intersection between a box $\mathcal{S}_b$ and a $(n-1)$-dimensional sphere $\mathcal{S}_p$, as follows:*

$$\mathbf{x} \in \{0,1\}^n \Leftrightarrow \mathbf{x} \in [0,1]^n \cap \left\{ \mathbf{x} : \left|\left| \mathbf{x} - \frac{1}{2}\mathbf{1}_n \right|\right|_p^p = \frac{n}{2^p} \right\}, \quad (2)$$

where $p \in (0, \infty)$, and $\mathcal{S}_b = [0,1]^n = \{\mathbf{x} | \ ||\mathbf{x}||_\infty \leq 1\}, \mathcal{S}_p = \{\mathbf{x} : ||\mathbf{x} - \frac{1}{2}\mathbf{1}_n||_p^p = \frac{n}{2^p}\}$. Note that $\mathcal{S}_p$ can be seen as a $(n-1)$-dimensional ℓ_p-sphere centered at $\frac{1}{2}\mathbf{1}_n$ with radius $\frac{n^{\frac{1}{p}}}{2}$.

Proof (Left $\Rightarrow$ Right). As $\{0,1\}^n \subset [0,1]^n$, given $\mathbf{x} \in \{0,1\}^n$, $\mathbf{x} \in [0,1]^n$ must hold. As $\{0,1\}^n \subset \{\mathbf{x} : ||\mathbf{x} - \frac{1}{2}\mathbf{1}_n||_p^p = \frac{n}{2^p}\}$, given $\mathbf{x} \in \{0,1\}^n$, $\mathbf{x} \in \{\mathbf{x} : ||\mathbf{x} - \frac{1}{2}\mathbf{1}_n||_p^p = \frac{n}{2^p}\}$ must hold. Combining these two points, we conclude: $\mathbf{x} \in [0,1]^n \cap \{\mathbf{x} : ||\mathbf{x} - \frac{1}{2}\mathbf{1}_n||_p^p = \frac{n}{2^p}\}$ must hold, given $\mathbf{x} \in \{0,1\}^n$.

(Right $\Rightarrow$ Left). As $\mathbf{x} \in [0,1]^n$, then $|x_i - \frac{1}{2}| \leq \frac{1}{2}\ \forall i$, and equality holds iff $x_i \in \{0,1\}$. As $p \in (0, \infty)$ and $|x_i - \frac{1}{2}| \leq \frac{1}{2}$, we have $|x_i - \frac{1}{2}|^p \leq \frac{1}{2^p}$, and equality holds iff $x_i \in \{0,1\}$. Then, we have $||\mathbf{x} - \frac{1}{2}\mathbf{1}_n||_p^p = \sum_i^n |x_i - \frac{1}{2}|^p \leq \frac{n}{2^p}$, and equality holds iff $x_i \in \{0,1\}\ \forall i$. Therefore, if $\mathbf{x} \in [0,1]^n \cap \{\mathbf{x} : ||\mathbf{x} - \frac{1}{2}\mathbf{1}_n||_p^p = \frac{n}{2^p}\}$, then $\mathbf{x} \in \{0,1\}^n$ must hold. □

Rather than adding these equivalent continuous constraints into the objective as penalty methods do, we embed them into the original problem by using the alternating direction method of multipliers (ADMM) [20]. In doing so, we introduce additional variables to separate the constraints, thus, simplifying the ADMM updates of all the primal variables without changing the form of the objective function, as formulated in Eq. (3). As we will describe in more detail later, $(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2)$ are updated in each ADMM step such that they move towards a local binary solution together, where $\mathbf{y}_1$ remains in the box, $\mathbf{y}_2$ on the shifted ℓ_p-sphere, and $\mathbf{x} \in \mathcal{C}$. Upon convergence, all three variables are equal and the resulting solution is binary

$$\min_{\mathbf{x},\mathbf{y}_1,\mathbf{y}_2} \ f(\mathbf{x}), \quad \text{s.t.} \begin{cases} \mathbf{x} \in \mathcal{C}, \mathbf{x} = \mathbf{y}_1, \mathbf{x} = \mathbf{y}_2 \\ \mathbf{y}_1 \in \mathcal{S}_b, \mathbf{y}_2 \in \mathcal{S}_p. \end{cases} \quad (3)$$

where $\mathcal{S}_b = \{\mathbf{y} : \mathbf{0} \leq \mathbf{y} \leq \mathbf{1}\}, \mathcal{S}_p = \{\mathbf{y} : ||\mathbf{y} - \frac{1}{2}\mathbf{1}||_p^p = \frac{n}{2^p}\}$.

Contributions. The contributions of the proposed ℓ_p-box ADMM method are four-fold. (i) To the best of our knowledge, this is the first work that uses the ℓ_p-box equivalence coupled with ADMM to solve IP problems. This combination enables a general optimization framework to solve these problems in the continuous domain by leveraging the flexibility and attractive properties of ADMM (e.g., simple sub-problems and aptitude for parallelization). We hope that this framework can serve as a basis for developing general-purpose or application-specific IP solvers. (ii) To focus our framework on some important applications in CV and ML, we target IP problems where $f(\mathbf{x})$ is quadratic and $\mathcal{C}$ is a polyhedron. In this case, the update steps are simple, as the most computationally expensive step is solving a positive definite linear system. (iii) We provide the theoretic analysis about the global convergence of the ℓ_p-box ADMM. By adding a perturbation with the sufficiently small factor ϵ to Problem (3), the globally converged solution generated by ℓ_p-box ADMM for the perturbed IP problem will be close to the stationary and feasible point of Problem (3) within $O(\epsilon)$. (iv) We apply the latter solver to three popular applications and compare it against widely used and state-of-the-art methods, some of which were specifically designed for the particular application. Extensive experiments show that our framework can efficiently produce state-of-the-art results.

2 Related Work

Integer programming has a very rich literature and a wide-range of developed methods and theory. In no way do we claim that we can give a detailed survey of all methods and variations of IP solvers here. However, in order to clarify the relationship and differences between our proposed ℓ_p-box ADMM method and existing ones, we group some widely used IP methods hierarchically (shown in Fig. 2) and discuss them briefly in what follows.

Discrete versus Continuous. Here, we distinguish between IP solvers that operate solely in the discrete domain and those that employ continuous optimization. Although IP is NP-hard in most cases, there do exist some discrete algorithms that guarantee the global solution in polynomial time for some particular IP forms. For example, if the IP is unconstrained and

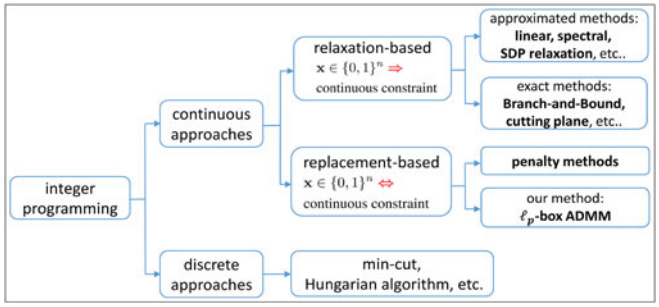

Fig. 2. A hierarchical organization of widely used integer programming (IP) methods.

$f(\mathbf{x})$ is submodular, then the global solution can be efficiently computed by the min-cut algorithm [21]. Another example is the Hungarian algorithm for the assignment problem [22]. However, there does not seem to be an efficient discrete method for the general constrained IP problem of Eq. (1). Discrete approaches are beyond the scope of this work, so we refer the readers to [23] for more specifically designed algorithms.

Since a practical discrete approach is not easy to find for general IP, much attention has been given to continuous approaches, owing to the advances in continuous optimization. The underlying motive behind this type of methods is to replace the binary constraints with continuous ones.

Relaxation versus Replacement. For continuous methods, the binary constraints are usually handled in one of two ways. The binary space can be relaxed to a larger continuous constraint space, thus, leading to *relaxation-based* approaches, or it can be replaced with an equivalent set of continuous constraints, thus, leading to *replacement-based* approaches, to which our proposed method belongs. In what follows, we give a brief overview of some popular examples of both types of continuous IP methods.

Relaxation Methods. They fall into two main categories: approximate and exact methods. Approximate methods usually optimize a continuous relaxed problem and then round the resulting continuous solution to generate a discrete one. Here, we briefly review three widely used forms of this type, including linear, spectral and semi-definite (SDP) relaxation. For the linear relaxation[1] method [25], the binary constraint is relaxed to the box constraint, i.e., $\mathbf{x} \in \mathcal{S}_b = [0,1]^n$. The main benefit here is in runtime because the simple box constraints can be efficiently embedded into continuous optimization solvers (e.g., interior-point) [26]. However, this relaxation is often too loose. Spectral relaxation [27] relaxes the binary constraint to the ℓ_2-ball, leading to a non-convex constraint. In SDP relaxation [28], [29], [30], the binary vector constraints are substituted with a positive semi-definite matrix constraint, i.e., $\mathbf{X} \in \mathbb{R}^{n \times n}$ and $\mathbf{X} \succeq 0$. Compared with linear and spectral relaxation, SDP relaxation is often tighter [5], [28], but with much higher memory and computation cost, despite the fact that there are recent efforts to alleviate these SDP side-effects [5], [31], [32]. Moreover, we realize that there are still many other types of relaxations and their variants in this sub-branch, such as quadratic [33], second-order-cone programming (SOCP) [34], and completely positive relaxation [35] to name a few. In general, a common drawback of approximate methods lies in the need to round/threshold the continuous solution to a binary one, which is not even guaranteed to be feasible. Since the optimization and rounding are performed separately, there is no guarantee that the obtained discrete solution is (locally) optimal in general. To obtain better discrete solutions, some exact relaxation methods have been developed, such as Branch-and-Bound [15] and cutting plane [16] methods. These methods call upon approximate methods in their sub-routines. Although BB and CP usually return feasible binary solutions (without the need for rounding), their common drawback is the slow runtime due to the repeated use of the linear relaxation.

Replacement Methods. They handle binary constraints by replacing them with equivalent continuous constraints. One popular group of these methods design specific penalty functions (non-convex in general) that are added to the objective $f(\mathbf{x})$, so as to encourage binary solutions. Conventional continuous techniques (e.g., interior-point methods) can then be applied to optimize this regularized problem at each iteration. These penalties are applied over and over again with increasing weights and generally guarantee convergence to feasible binary solutions. One drawback of such methods is that each iteration usually involves the minimization of a non-convex objective, which is difficult and time consuming in its own right, even if $f(\mathbf{x})$ is convex. Since the penalty function is increasingly enforced with more iterations, the non-convexity of the resulting optimization may lead to further issues, namely undesirable local minima and sensitivity to the initialization. Here, we note that recent efforts have been made to alleviate some of these issues [17], [18], [36]; however, they remain serious obstacles precluding the use of this type of solver in medium and large scale IP problems.

Our proposed method is also a replacement-based technique. Instead of adding a penalty to the objective, we use the ℓ_p-box equivalence in Eq. (1) within the ADMM framework to solve the equivalent problem in Eq. (3) without changing the objective. In this way, we separate the different constraints, leading to simple ADMM updates. Moreover, we inherit the attractive properties of ADMM, including granularity and aptitude for parallelization, which facilitate its use at large scales, as well as, for different types of objective $f(\mathbf{x})$ and constraint space $\mathcal{C}$.

Another related field is the literature of ADMM. Many variants of ADMM have been proposed, with the purpose of the better acceleration, convergence or stability. Some typical works include linearized ADMM, Bethe-ADMM, Bregman-ADMM. There are also some attempts to apply ADMM on some specific nonconvex problems [37], [38], [39], [40], [41]. However, ADMM is designed for continuous optimization, rather than integer programming. Although some variants of ADMM have been used to in some specific IP problems, such as MAP inference, actually they optimize the relaxation rather than the original IP problem. Our ℓ_p-box ADMM is a standard ADMM algorithm on nonconvex optimization. But any continuous optimization algorithm can be adopted, rather than only ADMM. Moreover, based on our equivalent continuous reformulation of the IP problem, we have built the bridge

1. Note that this relaxation is called as linear programming relaxation in Wikipedia. However, the linear programming relaxation also indicates a well-known MAP inference method [24]. To avoid confusion, we call this relaxation as linear relaxation in this manuscript.

between IP and the continuous optimization algorithms (including ADMM).

3 ℓ_p-Box ADMM

In this section, we give an overview of how ADMM can be used to solve the general IP problem in Eq. (3). First, we equivalently reformulate it into the following form:

$$\min_{\mathbf{x},\mathbf{y}} \; f(\mathbf{x}) + h(\mathbf{y}), \quad \text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{y}. \tag{4}$$

Here, $\mathbf{y} = [\mathbf{y}_1; \mathbf{y}_2]$ and $\mathbf{A} = [\mathbf{I}_n; \mathbf{I}_n] \in \{0,1\}^{2n \times n}$. $h(\mathbf{y}) = \mathbb{I}_{\{\mathbf{y}_1 \in \mathcal{S}_b \cap \mathcal{C}\}} + \mathbb{I}_{\{\mathbf{y}_2 \in \mathcal{S}_p\}}$. $\mathbb{I}_{\{a\}}$ denotes the indicator function: if a is $true$, then $\mathbb{I}_{\{a\}} = 0$, otherwise $\mathbb{I}_{\{a\}} = \infty$. This optimization is non-convex in general due to the ℓ_p-sphere constraint and possibly the nature of $\mathcal{C}$ and $f(\mathbf{x})$. Although ADMM has is widely used for convex (especially non-smooth) optimization [20], there has been growing interest and recent insights on the benefits of ADMM in non-convex optimization [37], [38], [39], [40], [41]. Inspired by this trend, we formulate the ADMM update steps for Eq. (3) based on the following augmented Lagrangian:

$$\begin{aligned}\mathcal{L}(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2, \mathbf{z}_1, \mathbf{z}_2) =& f(\mathbf{x}) + h(\mathbf{y}) + \mathbf{z}_1^\top (\mathbf{x} - \mathbf{y}_1) \\ &+ \mathbf{z}_2^\top (\mathbf{x} - \mathbf{y}_2) + \frac{\rho_1}{2}\|\mathbf{x} - \mathbf{y}_1\|_2^2 + \frac{\rho_2}{2}\|\mathbf{x} - \mathbf{y}_2\|_2^2.\end{aligned} \tag{5}$$

Here, $(\mathbf{z}_1, \mathbf{z}_2)$ indicate dual variables, while (ρ_1, ρ_2) are positive penalty parameters. Following the conventional ADMM process, we iteratively update the primal variables $(\mathbf{y}, \mathbf{x})$ by minimizing the augmented Lagrangian with respect to these variables, one at a time. Then, we perform gradient ascent on the dual problem to update $(\mathbf{z}_1, \mathbf{z}_2)$. Therefore, at iteration k, we perform the following update steps summarized in Algorithm 1.

Algorithm 1. General ℓ_p-Box ADMM Algorithm

Input: ADMM parameters and $\{\mathbf{x}^0, \mathbf{y}_1^0, \mathbf{y}_2^0, \mathbf{z}_1^0, \mathbf{z}_2^0\}$.
Output: $\mathbf{x}^*$.
While *not converged* **do**
  update $(\mathbf{y}_1^{k+1}, \mathbf{y}_2^{k+1})$ as shown in Eqs. (7) and (9)
  update $\mathbf{x}^{k+1}$ by solving Eq. (10)
  update $(\mathbf{z}_1^{k+1}, \mathbf{z}_2^{k+1})$ according to Eq. (11)
end
return $\mathbf{x}^* = \mathbf{x}^{k+1}$.

Update $\mathbf{y}_1^{k+1}$. The $\mathbf{y}_1$ sub-problem is reformulated as

$$\mathbf{y}_1^{k+1} = \arg\min_{\mathbf{y}_1 \in \mathcal{S}_b} (\mathbf{z}_1^k)^\top (\mathbf{x}^k - \mathbf{y}_1) + \frac{\rho_1}{2}\left\|\mathbf{y}_1 - \mathbf{x}^k\right\|_2^2. \tag{6}$$

It is a quadratic problem with a box constraint, which can be exactly solved by the following closed-form solution

$$\mathbf{y}_1^{k+1} = \mathbf{P}_{\mathcal{C}\cap\mathcal{S}_b}\left(\mathbf{x}^k + \frac{1}{\rho_1}(\mathbf{z}_1^k)^\top\right), \tag{7}$$

where the projection $\mathbf{P}_{\mathcal{C}\cap\mathcal{S}_b}(\mathbf{a})$ depends on the constraint space $\mathcal{C}$:

- If $\mathcal{C} = \mathbb{R}^n$, then it is the projection onto the $[0,1]^n$ box space, i.e., $\mathbf{P}_{\mathcal{S}_b}(\mathbf{a}) = \min(\mathbf{1}_n, \max(\mathbf{0}_n, \mathbf{a}))$.
- If $\mathcal{C} \cap \mathcal{S}_b$ is a probability simplex on partial or all dimensions of $\mathbf{y}_1$, then the projection can be exactly solved by the efficient algorithm presented in [42]. For example, $\mathbf{y}_1 \in \mathcal{C} \equiv \mathbf{1}_n^\top \mathbf{y}_1 = 1$.
- If $\mathcal{C} \cap \mathcal{S}_b$ are other forms of constraint spaces, such as a non-probability simplex (e.g., $\mathcal{C}$ is a general linear equality) or a half-space (e.g., $\mathcal{C}$ is a linear inequality), then it is difficult to obtain an exact solution. In this case, we suggest to move the constraint $\mathcal{C}$ onto $\mathbf{x}$, which will be presented in details in Section 4.

Update $\mathbf{y}_2^{k+1}$. The $\mathbf{y}_2$ sub-problem is reformulated as

$$\mathbf{y}_2^{k+1} = \arg\min_{\mathbf{y}_2 \in \mathcal{S}_p} (\mathbf{z}_2^k)^\top (\mathbf{x}^k - \mathbf{y}_2) + \frac{\rho_2}{2}\left\|\mathbf{y}_2 - \mathbf{x}^k\right\|_2^2. \tag{8}$$

It is a quadratic problem with a non-convex constraint $\mathcal{S}_p$, which requires the Euclidean projection onto $\mathcal{S}_p$. The complexity of such a projection depends on the value of p. For example, when $p = 2$, there is a closed-form solution, through a simple ℓ_2-normalization and shifting procedure. For the Euclidean projection onto other general ℓ_p spheres, we have not found much related work. Here, we propose to efficiently solve Problem (8) through two steps: optimize the unconstrained problem then project this solution onto the ℓ_p sphere, as follows:

$$\begin{cases} \mathbf{y}_2^{k+1} \;\; = \mathbf{P}_{\mathcal{S}_p}(\mathbf{x}^k + \frac{1}{\rho_2}\mathbf{z}_2^k), \\ \mathbf{P}_{\mathcal{S}_p}(\mathbf{a}) = \frac{n^{1/p}}{2}\frac{\bar{\mathbf{a}}}{\|\bar{\mathbf{a}}\|_p} + \frac{1}{2}\mathbf{1}_n, \quad \bar{\mathbf{a}} = \mathbf{a} - \frac{1}{2}\mathbf{1}_n. \end{cases} \tag{9}$$

It is easy to see that (a) the projection operator $\mathbf{P}_{\mathcal{S}_p}(\cdot)$ leads to a point on the ℓ_p sphere and (b) the above solution is optimal when $p = 2$. Although this solution is not optimal for other value of p, it is still a reasonable choice, at least owing to its efficient closed-form. This is also advocated in [20], where good performance on the original problem can be achieved even if the global solution for each sub-problem is not attained in each ADMM iteration. A detailed analysis of the influences of the projection operator $\mathbf{P}_{\mathcal{S}_p}(\cdot)$ with different p values will be presented in Section 3.2.

Update $\mathbf{x}^{k+1}$. This step requires solving the optimization in Eq. (10). Of course, the solution strategy is highly dependent on the nature of $f(\mathbf{x})$, which are application specific. Interestingly, when $f(\mathbf{x})$ is convex, then this update simply requires the evaluation of the proximal operator of $f(\mathbf{x})$ at $\frac{\rho_1\mathbf{y}_1^{k+1} + \rho_2\mathbf{y}_2^{k+1} - \mathbf{z}_1^k - \mathbf{z}_2^k}{\rho_1 + \rho_2}$. Moreover, when $f(\mathbf{x})$ is quadratic, then it is not difficult to see that $\mathbf{x}^{k+1}$ can be computed by solving a single linear system. In the next section, we will give a detailed treatment of how to update $\mathbf{x}^{k+1}$ when $f(\mathbf{x})$ is quadratic.

$$\arg\min_{\mathbf{x}} \; \frac{f(\mathbf{x})}{\rho_1 + \rho_2} + \frac{1}{2}\left\|\mathbf{x} - \frac{\rho_1\mathbf{y}_1^{k+1} + \rho_2\mathbf{y}_2^{k+1} - \mathbf{z}_1^k - \mathbf{z}_2^k}{\rho_1 + \rho_2}\right\|_2^2. \tag{10}$$

Update $(\mathbf{z}_1^{k+1}, \mathbf{z}_2^{k+1})$. We use conventional gradient ascent to update the dual variables

$$\begin{cases} \mathbf{z}_1^{k+1} = \mathbf{z}_1^k + \rho_1(\mathbf{x}^{k+1} - \mathbf{y}_1^{k+1}) \\ \mathbf{z}_2^{k+1} = \mathbf{z}_2^k + \rho_2(\mathbf{x}^{k+1} - \mathbf{y}_2^{k+1}). \end{cases} \tag{11}$$

3.1 Convergence Analysis

In this section we present the convergence analysis of the ℓ_p-box ADMM algorithm presented above. For clarity, we set $\rho = \rho_1 = \rho_2$.

3.1.1 Optimality Conditions of ℓ_p-Box ADMM Algorithm for (4)

We first check the optimality conditions of the variable sequence $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{z}^{k+1})$ generated by ADMM for Problem (4)

$$\begin{cases} \mathbf{z}^k + \rho(\mathbf{A}\mathbf{x}^k - \mathbf{y}^{k+1}) = \mathbf{z}^{k+1} - \rho\mathbf{A}(\mathbf{x}^{k+1} - \mathbf{x}^k) \\ \in \partial h(\mathbf{y}^{k+1}) \\ \nabla f(\mathbf{x}^{k+1}) + \mathbf{A}^\top \mathbf{z}^k + \rho\mathbf{A}^\top(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{y}^{k+1}) \\ = \nabla f(\mathbf{x}^{k+1}) + \mathbf{A}^\top \mathbf{z}^{k+1} = \nabla f(\mathbf{x}^{k+1}) + \mathbf{z}_1^{k+1} + \mathbf{z}_2^{k+1} = \mathbf{0}_n \\ \frac{1}{\rho}(\mathbf{z}^{k+1} - \mathbf{z}^k) = \rho(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{y}^{k+1}). \end{cases} \tag{12}$$

If there is a cluster point $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$, and $\mathbf{x}^*$ is a stationary point of (4), then the following conditions should be satisfied

$$\nabla f(\mathbf{x}^*) = -\mathbf{A}^\top \mathbf{z}^*,\ \mathbf{z}^* \in \partial h(\mathbf{y}^*),\ \mathbf{A}\mathbf{x}^* = \mathbf{y}^*. \tag{13}$$

3.1.2 ℓ_p-Box ADMM Algorithm for the Perturbed Problem of (4)

Here we study the convergence of ℓ_p-Box ADMM algorithm for optimizing Problem (4). We find the main difficulty during the convergence analysis is the constraint $\mathbf{A}\mathbf{x} = \mathbf{y}$, where $\mathbf{A} = [\mathbf{I}_n; \mathbf{I}_n]$ is full column rank, rather than full row rank. To tackle it, we propose to add a sufficient small perturbation onto the constraint, as follows:

$$\bar{\mathbf{A}}\bar{\mathbf{x}} = [\mathbf{A}, \epsilon\mathbf{I}_{2n}][\mathbf{x}; \hat{\mathbf{x}}] = \mathbf{A}\mathbf{x} + \epsilon\hat{\mathbf{x}} = \mathbf{y}, \tag{14}$$

where $\epsilon > 0$ is sufficient small, and $\hat{\mathbf{x}} \in \mathbb{R}^{2n}$ is an extra variable. Using this perturbed constraint, we present an associated perturbed problem of (4), as follows:

$$\min_{\bar{\mathbf{x}}, \mathbf{y}}\ \bar{f}(\bar{\mathbf{x}}) + h(\mathbf{y}),\ \text{ s.t. } \bar{\mathbf{A}}\bar{\mathbf{x}} = \mathbf{y}, \tag{15}$$

where $\bar{f}(\bar{\mathbf{x}}) = f(\mathbf{x}) + \frac{\epsilon}{2}\hat{\mathbf{x}}^\top\hat{\mathbf{x}}$, and $\bar{\mathbf{x}} = [\mathbf{x}; \hat{\mathbf{x}}] = [\mathbf{x}; \hat{\mathbf{x}}_1; \hat{\mathbf{x}}_2]$ with $\hat{\mathbf{x}}_1$, $\hat{\mathbf{x}}_2 \in \mathbb{R}^n$. Its augmented Lagrangian function is

$$\mathcal{L}_\rho(\bar{\mathbf{x}}, \mathbf{y}, \mathbf{z}) = \bar{f}(\bar{\mathbf{x}}) + h(\mathbf{y}) + \mathbf{z}^\top(\bar{\mathbf{A}}\bar{\mathbf{x}} - \mathbf{y}) + \frac{\rho}{2}\left\|\bar{\mathbf{A}}\bar{\mathbf{x}} - \mathbf{y}\right\|_2^2, \tag{16}$$

where $\mathbf{z} = [\mathbf{z}_1; \mathbf{z}_2]$ with $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^n$. The updates in ADMM algorithm are as follows:

$$\begin{cases} \mathbf{y}^{k+1} = \arg\min_{\mathbf{y}}\ h(\mathbf{y}) - (\mathbf{z}^k)^\top\mathbf{y} + \frac{\rho}{2}\left\|\bar{\mathbf{A}}\bar{\mathbf{x}}^k - \mathbf{y}\right\|_2^2 \\ \bar{\mathbf{x}}^{k+1} = \arg\min_{\bar{\mathbf{x}}}\ \bar{f}(\bar{\mathbf{x}}) + (\bar{\mathbf{A}}^\top\mathbf{z}^k)^\top\bar{\mathbf{x}} + \frac{\rho}{2}\left\|\bar{\mathbf{A}}\bar{\mathbf{x}} - \mathbf{y}^{k+1}\right\|_2^2 \\ \mathbf{z}^{k+1} = \mathbf{z}^k + \rho(\bar{\mathbf{A}}\bar{\mathbf{x}}^{k+1} - \mathbf{y}^{k+1}). \end{cases} \tag{17}$$

The optimality conditions of the variable sequence $(\bar{\mathbf{x}}^{k+1}, \mathbf{y}^{k+1}, \mathbf{z}^{k+1})$ are as follows:

$$\begin{cases} \mathbf{z}^k + \rho(\bar{\mathbf{A}}\bar{\mathbf{x}}^k - \mathbf{y}^{k+1}) = \mathbf{z}^{k+1} - \rho\bar{\mathbf{A}}(\bar{\mathbf{x}}^{k+1} - \bar{\mathbf{x}}^k) \\ \in \partial h(\mathbf{y}^{k+1}) \\ \nabla\bar{f}(\bar{\mathbf{x}}^{k+1}) + \bar{\mathbf{A}}^\top\mathbf{z}^k + \rho\bar{\mathbf{A}}^\top(\bar{\mathbf{A}}\bar{\mathbf{x}}^{k+1} - \mathbf{y}^{k+1}) \\ = \left[\nabla f(\mathbf{x}^{k+1}); \epsilon\hat{\mathbf{x}}^{k+1}\right] + \bar{\mathbf{A}}^\top\mathbf{z}^{k+1} \\ = \left[\nabla f(\mathbf{x}^{k+1}); \epsilon\hat{\mathbf{x}}^{k+1}\right] + \left[\mathbf{z}_1^{k+1} + \mathbf{z}_2^{k+1}; \epsilon\mathbf{z}_1^{k+1}; \epsilon\mathbf{z}_2^{k+1}\right] \\ = \mathbf{0}_{3n} \\ \frac{1}{\rho}(\mathbf{z}^{k+1} - \mathbf{z}^k) = \rho(\bar{\mathbf{A}}\bar{\mathbf{x}}^{k+1} - \mathbf{y}^{k+1}) \end{cases} \tag{18}$$

If there is a cluster point $(\bar{\mathbf{x}}^*, \mathbf{y}^*, \mathbf{z}^*)$, and $\bar{\mathbf{x}}^*$ is a stationary and feasible point of Problem (15), then the following stationary and feasibility conditions should be satisfied

$$[\nabla f(\mathbf{x}^*); \epsilon\hat{\mathbf{x}}^*] = -\bar{\mathbf{A}}^\top\mathbf{z}^*,\ \mathbf{z}^* \in \partial h(\mathbf{y}^*),\ \bar{\mathbf{A}}\bar{\mathbf{x}}^* = \mathbf{y}^*. \tag{19}$$

In the following we will prove that the ADMM algorithm for (15) globally converges to a stationary and feasible (i.e., integer) point, which is close to the stationary and feasible point of the original problem (4) within the range $O(\epsilon)$. As the perturbed problem (15) fully satisfies the assumptions of the problem studied in [40], we expand our convergence analysis by generally following the proof structure in [40].

3.1.3 Assumptions and Propositions

Assumption 1. *The objective function should satisfy the following conditions:*

- *f and h are semi-algebraic functions, and $h(\mathbf{y})$ is closed and proper*
- *There exist $\mathcal{Q}_1, \mathcal{Q}_2$ such that $\mathcal{Q}_1 \succeq \nabla^2_{\bar{\mathbf{x}}}[f(\mathbf{x}) + \epsilon\hat{\mathbf{x}}^\top\hat{\mathbf{x}}] \succeq \mathcal{Q}_2, \forall\bar{\mathbf{x}}$*
- *$\liminf_{\|\bar{\mathbf{x}}\|\to\infty} \|\nabla_{\bar{\mathbf{x}}}[f(\mathbf{x}) + \epsilon\hat{\mathbf{x}}^\top\hat{\mathbf{x}}]\| = \infty$*

Assumption 2. *The constraints and parameters should satisfy the following conditions:*

- *There exists $\sigma > 0$ such that $\bar{\mathbf{A}}\bar{\mathbf{A}}^\top \succeq \sigma\mathbf{I}_{2n}$*
- *$\mathcal{Q}_2 + \rho\bar{\mathbf{A}}^\top\bar{\mathbf{A}} \succeq \delta\mathbf{I}_{3n}$ for some $\rho, \delta > 0$, and $\rho \geq \frac{1}{\epsilon}$*
- *There exists $\mathcal{Q}_3 \succeq [\nabla^2 f(\mathbf{x}), \mathbf{0}_{2n}; \mathbf{0}_{2n}, \epsilon\mathbf{I}_{2n}]^2$, $\forall\mathbf{x}$, and $\delta\mathbf{I}_{3n} \succ \frac{2}{\sigma\rho}\mathcal{Q}_3$*
- *The constraint space $\mathcal{C}$ should be: $\mathbb{R}^n$; or, $\mathbf{y}_1 \in \mathcal{C} \cap \mathcal{S}_b$ is a probability simplex on partial or all dimensions of $\mathbf{y}_1$*
- *$p = 2$ in the ℓ_p projection (9)*

Remark on Assumptions. The definition of semi-algebraic functions can be found in [43], and it covers a broad class of functions, such as quadratic function, indicator function and cardinality function. The functions f and h considered in this work belong to this class. Since $h(\mathbf{y})$ in (15) is the summation of two indicator functions, it satisfies the first condition of Assumption 1. The second condition of Assumption 1 says that $f(\mathbf{x}) + \epsilon\hat{\mathbf{x}}^\top\hat{\mathbf{x}}$ should be twice differentiable and has a bounded Hessian matrix. If $f(\mathbf{x}) + \epsilon\hat{\mathbf{x}}^\top\hat{\mathbf{x}}$ is twice differentiable, the last condition must hold. In (15), $\bar{\mathbf{A}}$ is full row rank, thus the first condition in Assumption 2 holds. The second and third conditions can be satisfied by setting ρ sufficiently large. However, in later specific problems, we can derive a lower bound of ρ. The last two conditions on $\mathcal{C}$ and p are needed to ensure

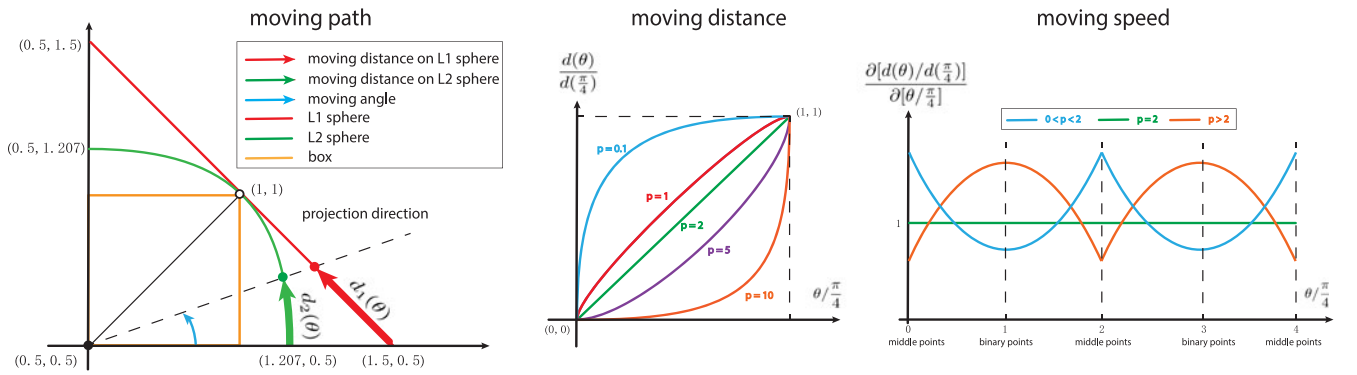

Fig. 3. Geometric illustration of the $\mathbf{z}_2$ update process using the ℓ_p projection. Left: The moving path of $\mathbf{z}_2$ through the ℓ_1 and ℓ_2 projection. Middle: Plots of the normalized moving distance $d(\theta)/d(\frac{\pi}{4})$ w.r.t. the normalized moving angle $\theta/\frac{\pi}{4}$ for different p values. Right: The moving speed of $\mathbf{z}_2$ at different places. See the text for a detailed description.

the exact solutions to the sub-problems w.r.t. $\mathbf{y}_1$ (see Eq. (6)) and $\mathbf{y}_2$ (see Eq. (8)) respectively, otherwise the optimality condition (see Eq. (18)) cannot be satisfied. However, if the condition on $\mathcal{C}$ cannot be satisfied, as suggested above, one can move $\mathcal{C}$ onto $\mathbf{x}$ as soft constraints, to ensure the exact solution for each sub-problem.

Proposition 2. *Suppose that Assumptions 1 and 2 hold, then the following conclusions are satisfied:*

- *The sequence* $\{(\bar{\mathbf{x}}^k, \mathbf{y}^k, \mathbf{z}^k)\}$ *generated from the ADMM algorithm has a cluster point* $(\bar{\mathbf{x}}^*, \mathbf{y}^*, \mathbf{z}^*)$
- $\{(\bar{\mathbf{x}}^k, \mathbf{y}^k, \mathbf{z}^k)\}$ *converges to* $(\bar{\mathbf{x}}^*, \mathbf{y}^*, \mathbf{z}^*)$
- $\bar{\mathbf{x}}^*$ *is a stationary and feasible point of (15), i.e., the stationary and feasibility conditions in (19) hold.*
- $\mathbf{z}^* \in \partial h(\mathbf{y}^*),\ \nabla f(\mathbf{x}^*) = -\mathbf{A}^\top \mathbf{z}^*,\ \|\mathbf{A}\mathbf{x}^* - \mathbf{y}^*\| \leq O(\epsilon)$

Following the proof structure in [40], we prove Proposition 2 with four steps, as follows:

1) We prove that if there is a variable sub-sequence generated by ADMM for Problem (15) that converges to a cluster point, then this cluster point satisfying the optimality conditions (19).
2) We prove that the whole variable sequence generated by ADMM is bounded, implying the existence of the cluster point, which is required in step 1).
3) We prove that the whole variable sequence generated by ADMM is globally convergent.
4) The limit of variable sequence generated by ADMM for Problem (15) will be close to the stationary and feasible point of Problem (4) within the range $O(\epsilon)$.

Due to space limitations, the detailed proof will be presented in *supplementary material*, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2018.2845842.

3.2 Understanding the ℓ_p Projection

In the proposed ℓ_p-box ADMM algorithm, the ℓ_p projection (i.e., the update of $\mathbf{y}_2$ in Eq. (9)) plays a key role, as it introduces the only non-convexity. We dedicate this section to understand how the chosen p value influences the optimization process and how it can be used to vary the (ρ_0, μ_ρ) parameters. In the experiments of the previous sections, we find that although the computational cost of the ℓ_p projection is the same for different p values, certain attributes of the optimization process (e.g., sensitivity to the ρ parameters and number of iterations for convergence) are significantly different.

To simplify the analysis and without loss of generality, we consider the projection onto the ℓ_1 and ℓ_2 spheres (i.e., $p \in \{1, 2\}$) in $\mathbb{R}^2$ in an example illustrated in Fig. 3. Vector space $\mathbb{R}^2$ is partitioned into 4 uniform areas: 4 quadrants with the shifted origin $(\frac{1}{2}, \frac{1}{2})$, represented by the degree range $(0, \frac{\pi}{2})$, $(\frac{\pi}{2}, \pi)$, $(\pi, \frac{3\pi}{2})$ and $(\frac{3\pi}{2}, 2\pi)$. Note that each quadrant includes one binary point. In Fig. 3 (*left*), we focus on the half-quadrant $(0, \frac{\pi}{4})$ that includes the single binary vector $(1, 1)$. Any point $\mathbf{a}$ in this area can be defined by a *moving angle* θ and its ℓ_1 and ℓ_2 projections are collinear with the center $(\frac{1}{2}, \frac{1}{2})$ (see the dots on red and green curves). In fact, all points $\mathbf{a}$ with the same θ have the same ℓ_p projections. During the optimization of our ℓ_p-box ADMM algorithm, $\mathbf{y}_2$ is encouraged to gradually move along its corresponding ℓ_p-sphere from $\theta = 0$ towards $\theta = \frac{\pi}{4}$, where the binary solution is. For example, if using the ℓ_1 projection, $\mathbf{y}_2$ will move along the straight segment between $(1.5, 0.5)$ and $(1, 1)$; if using the ℓ_2 projection, $\mathbf{y}_2$ will move along the arc between $(1.207, 0.5)$ and $(1, 1)$.

We define a *moving distance* $d_p(\theta)$ to evaluate how close $\mathbf{y}_2$ is to the binary solution along the ℓ_p-sphere. This can give us a new interpretation for the ℓ_p projection. To do this, we define the normalized angle $\overline{\theta} = \theta/\frac{\pi}{4}$ and the normalized distance $\overline{d}_p(\overline{\theta}) = d_p(\theta)/d_p(\frac{\pi}{4})$. The relationship between $\overline{d}_p(\overline{\theta})$ and $\overline{\theta}$ is as follows.

- For $p = 1$, $\overline{d}_1(\overline{\theta}) = \frac{2\tan(\frac{\pi}{4}\overline{\theta})}{1+\tan(\frac{\pi}{4}\overline{\theta})} \in [0, 1]$
- For $p = 2$, $\overline{d}_2(\overline{\theta}) = \overline{\theta} \in [0, 1]$,

where $d_1(\frac{\pi}{4}) = \frac{\sqrt{2}}{2}$ and $d_2(\frac{\pi}{4}) = \frac{\sqrt{2}\pi}{8}$. Note that the subscript corresponds to the p value. As shown in Fig. 3 (*middle*), we plot the curves of $(\overline{\theta}, \overline{d}_p(\overline{\theta}))$ for different p values. Note that for p values excluding $\{1, 2\}$, where we do not have simple formulations for $\overline{d}(\overline{\theta})$, we generate the plots numerically. Furthermore, it is easy to compute the angular *moving speed* of $\mathbf{y}_2$ for $p = \{1, 2\}$ as follows,

- For $p = 1$, $\frac{\partial \overline{d}_1(\overline{\theta})}{\partial \overline{\theta}} = \frac{\pi}{2}\frac{1+\tan^2(\overline{\theta}\frac{\pi}{4})}{(1+\tan(\overline{\theta}\frac{\pi}{4}))^2} \in [\frac{\pi}{4}, \frac{\pi}{2}]$.
- For $p = 2$, $\frac{\partial \overline{d}_2(\overline{\theta})}{\partial \overline{\theta}} = 1$.

The moving speeds for different values of $\overline{\theta}$ across different ranges of p are plotted in Fig. 3 (*right*). Based on these plots in $\mathbb{R}^2$, we can make the following observations:

1) When $p = 2$, $\mathbf{y}_2 = \mathbf{P}_{\mathcal{S}_p}(\mathbf{a})$ always moves with unit speed no matter where $\mathbf{a}$ is in the half quadrant.
2) When $p \in (0, 2)$: if $\overline{\theta} = 0$ (i.e., $\mathbf{a}$ is located at the middle place between two binary points), its speed is higher than 1, and smaller p values correspond to higher speeds; if $\overline{\theta} = \frac{\pi}{4}$ (i.e., $\mathbf{a}$ is collinear to the binary point and the center), its speed is lower than 1, and smaller p values correspond to smaller speeds. This means that when using a smaller p, $\mathbf{y}_2$ is more sensitive to the change of $\mathbf{a}$ when it is in between binary points, while being less sensitive around binary points. If we use a binary initialization for $\mathbf{y}_2$ and $\mathbf{a}$ (we do this in clustering and matching experiments), then the optimization process of our method can be intuitively seen as doing the following: $\mathbf{y}_2$ first jumps from one quadrant to other quadrants, then stays in one area, and finally converges to the corresponding binary point. During this process, $\mathbf{y}_2$ can easily jump between different areas, leading to the frequent oscillation of the objective curve. To alleviate the oscillation, empirical evidence suggests that ρ_0 or μ_ρ be decreased, leading to smaller changes in $\mathbf{a}$. This trick is very useful in our experiments.
3) When $p \in (2, +\infty)$: if $\overline{\theta} = 0$, its speed is lower than 1 and smaller p values have smaller speeds; if $\overline{\theta} = \frac{\pi}{4}$, its speed is higher than 1, and smaller p values have higher speeds. This means that when using a larger p, $\mathbf{y}_2$ is less sensitive to the change of $\mathbf{a}$ when it is in between binary points, while being more sensitive around binary points. During the optimization process, in early iterations, the change of $\mathbf{y}_2$ and $\mathbf{a}$ tend to be large. But, when they approach to the middle place, $\mathbf{y}_2$ gets slower and it is difficult to jump to other quadrants (corresponding to other binary points). Thus, the expected shape of the objective curve is expected to be quickly decreasing in the early iterations with a long smooth tail. A side-effect of this situation is that its performance is more dependent on the initialization, and it may converge to a poor solution. To alleviate this drawback, our experience is to increase ρ_0 or μ_ρ, leading to larger changes in $\mathbf{a}$. This trick is very useful in our experiments.

Although above observations are made in $\mathbb{R}^2$, they are naturally extended to higher dimensional spaces $\mathbb{R}^n$.

4 ℓ_p-Box ADMM for BQPs

In this section, we focus on the popular BQP problem in Eq. (20). Without loss of generality, we assume that $\mathbf{L} \succeq 0$. This is valid because $\mathbf{x}^\top\mathbf{x} = \mathbf{1}^\top\mathbf{x}$ when $\mathbf{x} \in \{0,1\}^n$, and thus, $\mathbf{x}^\top\mathbf{M}\mathbf{x} = \mathbf{x}^\top(\mathbf{M} + \alpha\mathbf{I})\mathbf{x} - \alpha\mathbf{1}^\top\mathbf{x}$ for any α and $\mathbf{M}$

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}_1,\mathbf{y}_2} \quad & f(\mathbf{x}) = \mathbf{x}^\top\mathbf{L}\mathbf{x} + \mathbf{b}^\top\mathbf{x} \\ \text{s.t.} \quad & \begin{cases} \mathbf{x} \in \mathcal{C} = \{\mathbf{x} : \mathbf{C}_1\mathbf{x} = \mathbf{d}_1; \mathbf{C}_2\mathbf{x} \le \mathbf{d}_2\} \\ \mathbf{y}_1 \in \mathcal{S}_b; \quad \mathbf{y}_2 \in \mathcal{S}_p; \quad \mathbf{x} = \mathbf{y}_1 = \mathbf{y}_2. \end{cases} \end{aligned} \tag{20}$$

We can invoke Algorithm 1 to solve the BQP. The $\mathbf{x}$ subproblem is actually a convex QP with linear constraints, whose global solution can be computed with many iterative off-the-shelf QP solvers (e.g., as simple as the built-in function *quadprog* in MATLAB). In practice, these methods can be computationally expensive especially at large-scales. Since we need to solve this kind of optimization in every ADMM iteration, we prefer to have the per-iteration cost to be as low as possible. In what follows, we will show that by adding more variables to the problem, the most complicated update step becomes solving a single positive-definite linear system. To do this, we add an auxiliary variable $\mathbf{y}_3$ to transform $\{\mathbf{C}_2\mathbf{x} \le \mathbf{d}_2\}$ in Eq. (20) into $\{\mathbf{C}_2\mathbf{x} + \mathbf{y}_3 = \mathbf{d}_2;\ \mathbf{y}_3 \in \mathbb{R}^n_+\}$.

Following the same ADMM strategy, we use two more dual variables $(\mathbf{z}_3, \mathbf{z}_4)$ for the linear equalities and inequalities and formulate the following augmented Lagrangian

$$\begin{aligned} \mathcal{L} = & f(\mathbf{x}) + g_1(\mathbf{y}_1) + g_2(\mathbf{y}_2) + g_3(\mathbf{y}_3) + \mathbf{z}_1^\top(\mathbf{x} - \mathbf{y}_1) \\ & + \mathbf{z}_2^\top(\mathbf{x} - \mathbf{y}_2) + \mathbf{z}_3^\top(\mathbf{C}_1\mathbf{x} - \mathbf{d}_1) + \mathbf{z}_4^\top(\mathbf{C}_2\mathbf{x} + \mathbf{y}_3 - \mathbf{d}_2) \\ & + \frac{\rho_1}{2}\|\mathbf{x} - \mathbf{y}_1\|_2^2 + \frac{\rho_2}{2}\|\mathbf{x} - \mathbf{y}_2\|_2^2 + \frac{\rho_3}{2}\|\mathbf{C}_1\mathbf{x} - \mathbf{d}_1\|_2^2 \\ & + \frac{\rho_4}{2}\|\mathbf{C}_2\mathbf{x} + \mathbf{y}_3 - \mathbf{d}_2\|_2^2, \end{aligned} \tag{21}$$

where $g_1(\mathbf{y}_1) = \mathbb{I}_{\{\mathbf{y}_1 \in \mathcal{S}_b\}}$, $g_2(\mathbf{y}_2) = \mathbb{I}_{\{\mathbf{y}_2 \in \mathcal{S}_p\}}$, $g_3(\mathbf{y}_3) = \mathbb{I}_{\{\mathbf{y}_3 \in \mathbb{R}^n_+\}}$. The update steps for $(\mathbf{y}_1, \mathbf{y}_2,\ \mathbf{z}_1, \mathbf{z}_2)$ are exactly the same as in Algorithm 1. The only additional steps that are needed involve $(\mathbf{x}, \mathbf{y}_3, \mathbf{z}_3, \mathbf{z}_4)$. We summarize them next.

Update $\mathbf{x}^{k+1}$. This step requires the minimization of a strongly convex QP without constraints. By setting the gradient to zero, we can compute $\mathbf{x}^{k+1}$ by solving the following positive-definite linear system. This can be done efficiently using the preconditioned conjugate gradient (PCG) method, especially for large sparse matrices

$$\begin{aligned} & \big(2\mathbf{L} + (\rho_1 + \rho_2)\mathbf{I} + \rho_3\mathbf{C}_1^\top\mathbf{C}_1 + \rho_4\mathbf{C}_2^\top\mathbf{C}_2\big)\mathbf{x}^{k+1} \\ & = \rho_1\mathbf{y}_1^{k+1} + \rho_2^k\mathbf{y}_2^{k+1} + \rho_3\mathbf{C}_1^\top\mathbf{d}_1 + \rho_4\mathbf{C}_2^\top(\mathbf{d}_2 - \mathbf{y}_3^{k+1}) \\ & \quad - \mathbf{b} - \mathbf{z}_1^k - \mathbf{z}_2^k - \mathbf{C}_1^\top\mathbf{z}_3^k - \mathbf{C}_2^\top\mathbf{z}_4^k. \end{aligned} \tag{22}$$

Update $(\mathbf{y}_3^{k+1}, \mathbf{z}_3^{k+1}, \mathbf{z}_4^{k+1})$. These are simple updates. The projection onto $\mathbb{R}^n_+$ is an element-wise truncation at 0

$$\begin{cases} \mathbf{y}_3^{k+1} = \mathbf{P}_{\mathbb{R}^n_+}\left(\mathbf{d}_2 - \mathbf{C}_2\mathbf{x}^k - \frac{\mathbf{z}_4^k}{\rho_4}\right) \\ \mathbf{z}_3^{k+1} = \mathbf{z}_3^k + \gamma\rho_3(\mathbf{C}_1\mathbf{x}^{k+1} - \mathbf{d}_1) \\ \mathbf{z}_4^{k+1} = \mathbf{z}_4^k + \gamma\rho_4(\mathbf{C}_2\mathbf{x}^{k+1} + \mathbf{y}_3^{k+1} - \mathbf{d}_2). \end{cases} \tag{23}$$

Implementation Details. For convenience, we set all the ρ parameters to the same value in the same iteration. At the end of each ADMM iteration, ρ is increased by $\mu_\rho\%$ with $\mu_\rho \in [0, 5]$. We also set a upper limit ρ_{max}, such that $\rho \leftarrow \min(\rho\mu_\rho, \rho_{max})$. We experimented with different stopping criteria (e.g., relative change in $\mathbf{x}$, difference between $\mathbf{x}$ and $(\mathbf{y}_1, \mathbf{y}_2)$, and relative change in continuous objective), all of which have comparable performance. In the reported experiments, we adopt these criteria jointly, which means that the algorithm will stop if all criteria are satisfied. Specifically, we set the same threshold value 0.0001 for all criteria. Furthermore, all the $\mathbf{y}$ and $\mathbf{z}$ variables are initialized to $\mathbf{0}$, while $\mathbf{x}$ is initialized as a random binary vector, or by using the results of other methods (e.g., using K-means as initialization in clustering). The computational complexity

of the ℓ_p-box ADMM is linear in n, i.e., $\mathcal{O}(T_{admm}(T_{pcg}n_{|\mathbf{A}|} + an))$, with T_{admm} and T_{pcg} ($T_{pcg} < 10$ in our experiments) being the iterations of the ADMM and PCG respectively, a is a small scalar (depending on the number of rows of $\mathbf{C}_1$ and $\mathbf{C}_2$), and $n_{|\mathbf{A}|}$ indicating the number of non-zero entries in $\mathbf{A}$. When $\mathbf{A}$ is sparse (which is indeed the case in many real problems), $n_{|\mathbf{A}|}$ is linear in n. Our algorithm is implemented in MATLAB.

Remark on Convergence. As $\mathbf{L} \succeq 0$, $f(\mathbf{x})$ satisfies the conditions in Assumption 1. Moreover, as demonstrated under Eq. (7), if $\mathbf{C}_1, \mathbf{d}_1$ don't exist, or $\{\mathbf{y}_1 \,|\, \mathbf{C}_1\mathbf{y}_1 = \mathbf{d}_1\} \cap \mathcal{S}_b$ is a probability simplex on partial or all dimensions of $\mathbf{y}_1$, as well as that $\mathbf{C}_2, \mathbf{d}_2$ don't exist, then the sub-problem with respect to $\mathbf{y}_1$ (see Eq. (6)) can be solved exactly. Consequently, we can move the equality constraint $\mathbf{C}_1\mathbf{y}_1 = \mathbf{d}_1$ onto $\mathbf{y}_1$, then the global convergence presented in Proposition 2 holds for ℓ_p-box ADMM for BQPs. On the other hand, if $\mathbf{C}_2, \mathbf{d}_2$ exist, or $\{\mathbf{y}_1 \,|\, \mathbf{C}_1\mathbf{y}_1 = \mathbf{d}_1\} \cap \mathcal{S}_b$ isn't a probability simplex, then the theoretic global convergence can not be guaranteed. In this case, we suggest to move both equality and inequality constraints onto $\mathbf{x}$, and handle them as presented above in this section. In our experiments, the ADMM algorithm still always gives the converged integer solution.

5 Pairwise MRF

5.1 Formulation

Given a Markov Random Field (MRF) model, which is constructed based on a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with $\mathcal{V}$ being a set of n nodes and $\mathcal{E}$ being the edge set, the energy pairwise MRF minimization problem is generally formulated as follows [45]:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbb{E}(\mathbf{x}) = \mathbf{x}^\top \mathbf{L}\mathbf{x} + \mathbf{d}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{C}_1\mathbf{x} = \mathbf{1}; \quad \mathbf{x} \in \{0,1\}^{nK\times 1}, \end{aligned} \tag{24}$$

where $\mathbf{x}$ is a concatenation of all indicator vectors for the states $k \in \{1, \ldots, K\}$ and all n nodes. For example, if $\mathbf{x}_k^i = 1$, then node i takes on the state k; otherwise, $\mathbf{x}_k^i = 0$. Since each node can only take on one state, we enforce that $\sum_{k=1}^{K} \mathbf{x}_k^i = \mathbf{1}$ for $\forall i \in \mathcal{V}$, which is formulated as a sparse linear system of equalities: $\mathbf{C}_1\mathbf{x} = \mathbf{1}$. Here, $\mathbf{L} \in \mathbb{R}^{nK\times nK}$ is the un-normalized Laplacian of $\mathcal{G}$, i.e., $\mathbf{L} = \mathbf{D} - \mathbf{W}$ with $\mathbf{W}$ being the matrix of node-to-node similarities. Our ℓ_p-box ADMM algorithm in Section 4 can be used to solve Eq. (24). Interestingly, practical graph constraints can be embedded into Eq. (24) as linear constraints, such as hard (some nodes should have a particular state), mutually exclusive (some nodes should have different states) and cardinality (a limit on the number of nodes belonging to a particular state) constraints.

Popular Methods. It has been proven that when $K = 2$, Eq. (24) is a submodular minimization problem, and it can be globally optimized using the min-cut algorithm (a discrete method) in polynomial time [21], [46]. However, when $K > 2$, this global solution cannot be guaranteed in general.

5.2 Image Segmentation Experiments

Here, we target the energy minimization problem of Eq. (24) applied to binary and multi-class image segmentation.

Experimental Setup. We compare our method against three generic IP solvers, namely linear relaxation, spectral relaxation and an exact penalty method [17], as well as, a state-of-the-art and widely used min-cut implementation [47]. A state-of-the-art MAP inference method called Bethe ADMM [44] which is formulated based on LP relaxation [24] is also compared. SDP relaxation is not feasible in this scenario because the number of variables is n^2, where n is the number of pixels in the image. The linear relaxation method is implemented using the built-in function *quadprog* in MATLAB. In terms of the spectral relaxation, its closed-form solution based on eigen-decomposition of is also implemented using MATLAB. Bethe-ADMM is implemented by C language.[2] We follow the typical setup in graph-based image segmentation. The similarity matrix is defined on an 8-pixel neighborhood and each element $\mathbf{W}_{ij} = \exp(-\|\mathbf{c}_i - \mathbf{c}_j\|_2^2)$, where $\mathbf{c}_i$ is the RGB color of pixel i. The user indicates pixels that belong to each state, by drawing a color-coded brush stroke for each state on the image. The unary costs $\mathbf{d}$ are computed from the negative log-likelihood of pixels in the image belonging to each of the K states. We initialize the linear relaxation, penalty, and our method using a uniformly random label image.

Comparison. First, we compare all methods in terms of their final energy value and runtime in the case of binary submodular MRF ($K = 2$). Here, we consider the *cameraman* image at different sizes: $n = \{10^3, \ldots, 5 \times 10^6\}$ and repeat each segmentation five times. We summarize the mean and std values of the objective and runtime in Table 1. Since the penalty method needs to solve many increasingly non-convex problems, its slow runtime makes it infeasible for larger sized images. Clearly, our method with different ℓ_p projections achieve energies that are very close to the global minimum (min-cut result), far closer than other IP methods excluding Bethe-ADMM. Bethe ADMM gives slightly lower energy than ℓ_p-box ADMM, while its runtime is much higher than that of ℓ_p-box ADMM. Bethe ADMM is specially designed for MAP inference in submodular MRF, and its main technique is replacing the ℓ_2 penalty term in the augmented Lagrangian function of ADMM by the Bethe entropy. This technique can be naturally adopted in ℓ_p-box ADMM algorithm. Note that we adopt the same parameters for different ℓ_p projections, thus, demonstrating that our method is robust to different p values. The approach to do the parameter adjustment for different p values will be presented in Section 3.2. Interestingly, our std values are much lower than the penalty method, which indicates that our method is less sensitive to the initialization and is less prone to getting stuck in undesirable local minima, which is a major issue in non-convex optimization in general. Note that the linear relaxation method has a zero standard deviation energy because the convexity of the relaxed problem guarantees convergence to the same solution no matter the initialization. Moreover, our method exhibits a runtime that is $\mathcal{O}(n)$ owing to the fact that the number of non-zero elements in $\mathbf{L}$ is $8n$. It converges considerably faster than the other IP solvers.

In Fig. 4, we validate our convergence guarantee by showing the continuous solution $\mathbf{x}$ in sample ADMM iterations. In only 25 iterations, the randomly initialized solution reaches an almost binary state, whose energy is close to the global minimum. Upon convergence (iteration 120), the final solution is binary and its energy is only 0.4 percent larger than the

2. Downloaded from "http://www-users.cs.umn.edu/~qifu/"

TABLE 1
Energy Minimization Results on *Cameraman* Showing Mean(std) Values of the Energy of Eq. (24) and Runtime (Seconds) for All Methods Over 5 Runs

size →	$n = 10^3$		$n = 5 \times 10^3$		$n = 10^4$		$n = 5 \times 10^4$	
method ↓	energy	runtime	energy	runtime	energy	runtime	energy	runtime
min-cut [21]	−163(0)	4e-3(0)	−1372(0)	7e-3(0)	−3228(0)	0.02(0)	−20481(0)	0.07(0)
linear relaxation	−108(0)	0.09(0)	−1319(0)	0.4(0)	−2890(0)	0.80(0)	−19693(0)	5.22(0.2)
spectral relaxation [27]	206(0)	0.71(0)	−415(0)	1.41(0)	−1242(0)	0.11(0)	−8809(0)	1.67(0)
Bethe-ADMM [44]	−163(0)	0.38(0)	−1372(0)	2.47(0)	−3228(0)	5.05(0)	−20466(0)	24.78(0)
penalty [17]	−157(7)	84(23)	−1325(12)	963(85)	N/A	N/A	N/A	N/A
$\ell_{0.5}$-box ADMM	−163(0)	0.08(0)	−1372(0)	0.23(0)	−3219(0)	0.43(0)	−20380(0)	2.37(0.02)
ℓ_1-box ADMM	−163(0)	0.08(0)	−1372(0)	0.23(0.01)	−3222(0)	0.39(0.01)	−20396(0)	2.14(0.1)
ℓ_2-box ADMM	−162(0)	0.07(0)	−1372(0)	0.19(0)	−3218(0)	0.33(0)	−20390(0)	1.84(0.01)
ℓ_5-box ADMM	−162(0)	0.08(0)	−1372(0)	0.28(0)	−3217(0)	0.49(0.01)	−20379(0)	2.77(0.03)
ℓ_{10}-box ADMM	−162(0)	0.08(0)	−1372(0)	0.26(0)	−3213(0)	0.46(0.01)	−20323(0)	2.58(0.01)
size →	$n = 10^5$		$n = 5 \times 10^5$		$n = 10^6$		$n = 5 \times 10^6$	
method ↓	energy	runtime	energy	runtime	energy	runtime	energy	runtime
min-cut [21]	−43711(0)	0.13(0)	−254672(0)	0.74(0)	−537557(0)	2.59(0)	−2959932(0)	11.47(0)
linear relaxation	−42530(0)	12.16(0.1)	−177932(0)	127.0(0.85)	−429694(0)	669.6(1.1)	N/A	N/A
spectral relaxation [27]	−22911(0)	1.47(0)	−200645(0)	1.93(0)	−429694(0)	3.26(0)	−2844120(0)	5.12(0)
Bethe-ADMM [44]	−43694(0)	32.72(0)	−254641(0)	183.23(0)	−537090(0)	390.45(0)	−2959166(0)	1769.62(0)
$\ell_{0.5}$-box ADMM	−43547(0)	5.03(0.16)	−253252(0)	27.77(0.06)	−532998(0)	49.56(0.14)	−2929903(0)	261.34(0.25)
ℓ_1-box ADMM	−43614(0)	4.25(0.02)	−253747(0)	24.5(0.13)	−534317(0)	46.15(0.06)	−2942986(0)	216.89(0.23)
ℓ_2-box ADMM	−43611(0)	4.02(0.005)	−253717(0)	22.96(0.08)	−534261(0)	41.15(0.02)	−2932654(0)	209.72(0.14)
ℓ_5-box ADMM	−43536(0)	5.49(0.01)	−253169(0)	30.96(0.05)	−533150(0)	55.52(0.08)	−2936757(0)	248.21(0.12)
ℓ_{10}-box ADMM	−43432(0)	5.46(0.02)	−252282(0)	28.25(0.19)	−529809(0)	53.64(0.03)	−2929080(0)	248.65(0.09)

min-cut result. Finally, we show qualitative segmentation results of our method and min-cut in Fig. 5 for different versions of the image segmentation problem, including GrabCut [14], interactive segmentation [48], and multi-class segmentation [49].

6 Graph Matching

6.1 Formulation

The general formulation of graph matching is [9]

$$\max_{\mathbf{x}\in\{0,1\}^n} \mathbf{x}^\top \mathbf{M}\mathbf{x} \quad \text{s.t.} \quad \mathbf{C}_2\mathbf{x} \leq 1, \tag{25}$$

where $\mathbf{x}$ is an indicator vector with $\mathbf{x}_{ia} = 1$ if node i from the first graph (e.g., feature i in one image) is matched to node a from the other graph (e.g., feature a in another image) and 0 otherwise. The constraint $\mathbf{C}_2\mathbf{x} \leq 1$ enforces the one-to-one constraint in matching. Here, $\mathbf{C}_2 = [\mathbf{1}_{n_2}^\top \otimes \mathbf{I}_{n_1}; \mathbf{I}_{n_2} \otimes \mathbf{1}_{n_1}^\top] \in \{0,1\}^{(n_1+n_2)\times n}$ with n_1 and n_2 being the number of nodes in the two graphs and $n = n_1 n_2$. The Kronecker product is denoted by $\otimes$.

BQP Reformulation. As demonstrated in [8], the non-negative similarity matrix $\mathbf{M}$ is rarely positive semi-definite in real matching problems. But, we can easily transform Eq. (25) into the BQP form in Eq. (20), by employing a simple trick in binary programming: $\mathbf{x}^\top\mathbf{M}\mathbf{x} = \mathbf{x}^\top(\mathbf{M} - \mathbf{D})\mathbf{x} + \mathbf{d}^\top\mathbf{x} = -\mathbf{x}^\top\mathbf{L}\mathbf{x} + \mathbf{d}^\top\mathbf{x}$, where $\mathbf{d} = \mathbf{M}\mathbf{1}$ is the degree vector, $\mathbf{D} = diag(\mathbf{d})$ is the degree matrix, and $\mathbf{L} \succeq 0$ is the resulting Laplacian matrix. As such, we form the equivalent problem

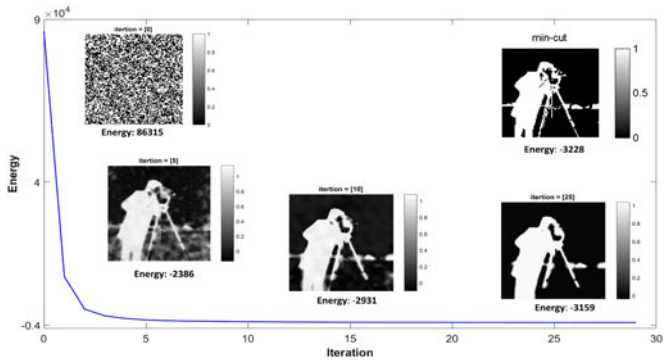

Fig. 4. The optimization procedure of ℓ_2-box ADMM on the segmentation task of the *cameraman* image (resized to 100×100, i.e., $n = 1e4$). Notice how the continuous solution becomes more binary and how the energy decreases, getting closer to the global minimum (min-cut result). At convergence (iteration 120), the final solution is binary and its energy is only 0.4 percent away from the global minimum.

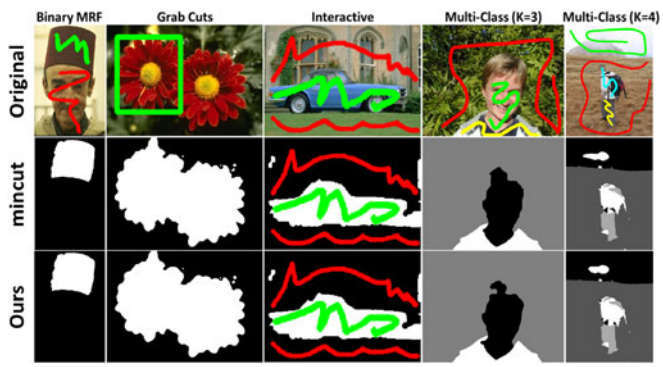

Fig. 5. MRF-based segmentation results. The user can determine the unary costs using free-hand strokes (1st column) or a simple rectangle similar to [14] (2nd column). The user can add hard linear constraints (3rd column) by identifying pixels belonging to the foreground (green) or background (red). In the 4th and 5th columns, we show some multi-class results with $K = 3, 4$. Note how our ℓ_p-box method converges to discrete solutions that are very similar to those of min-cut, a state-of-the-art application-specific algorithm.

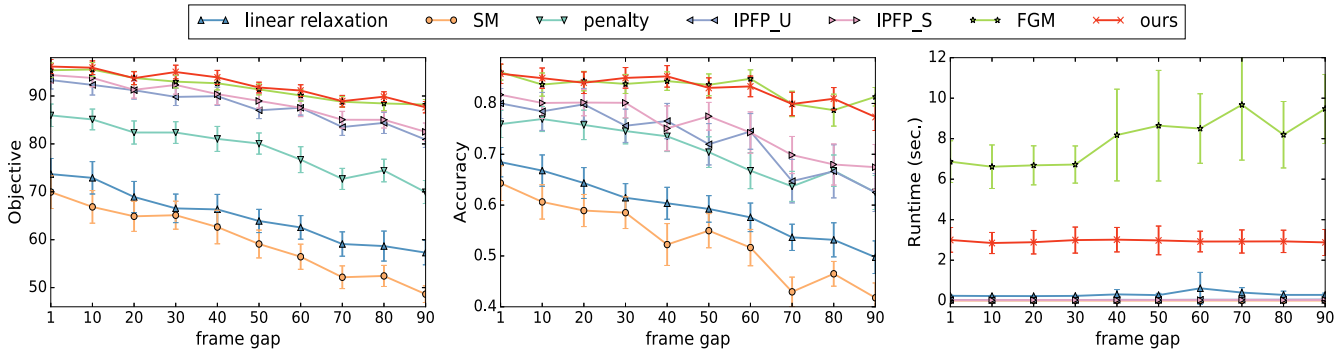

Fig. 6. Graph matching results. For visualization purposes, the error bars of the objective and accuracy (std) are made smaller ($\frac{1}{5}$) for all methods. The runtime of the penalty method (311 ± 23 seconds) is much higher than other methods, thus it is not shown here. Since ℓ_p-box ADMM with different p values (we use $p = 0.5, 1, 2, 5, 10$) converge to the same objective and accuracy, with slight differences in runtime, we only show the results of ℓ_2-box ADMM for simplicity.

$$\min_{\mathbf{x}\in\{0,1\}^n} \mathbf{x}^\top \mathbf{L}\mathbf{x} - \mathbf{d}^\top \mathbf{x}, \quad \text{s.t.} \quad \mathbf{C}_2\mathbf{x} \leq \mathbf{1}. \tag{26}$$

Popular Methods. Many methods have been specifically designed to solve the above matching problem. For example, the integer projected fixed point (IPFP) method [8] iteratively determines a search direction in the discrete domain to update the continuous solution of the unconstrained problem. Its major drawback is that it does not guarantee convergence to a feasible binary solution [50]. A more recent method called factorized graph matching (FGM) [9] combines a convex and a concave relaxation, by utilizing the fact that the matrix $\mathbf{M}$ can be decomposed into smaller matrices. Then, the Frank-Wolfe (FW) [51] algorithm is adopted to optimize the relaxed problem. Although FGM guarantees a feasible binary solution, it is costly due to the repeated use of FW.

6.2 Matching Experiments on a Video Sequence

We test the matching problem in Eq. (25) on a benchmark sequence called *house* [9], comprising 111 frames.

Experimental Setup. Our method is compared against two generic IP solvers,[3] namely the linear relaxation and an exact penalty method [17], as well as, several state-of-the-art methods, namely SM (uses spectral relaxation) [54], IFPU$_U$ (using a doubly stochastic matrix as the initialization), IFPU$_S$ (using the solution of SM as the initialization) [8]. Note that the matching results of SM are used to initialize IFPU$_S$, FGM, and our method. The implementations of SM, IFPU$_U$, IFPU$_S$ and FGM are freely downloaded from *http://www.f-zhou.com/gm.html*. We adopt the same settings as [9], including the nodes, edges, features, and edge similarities (refer to [9] for more details). Specifically, 30 landmark points are detected in each frame, from which 5 points are randomly picked and removed. Each pair of frames with a fixed frame gap are matched. For example, if the frame gap is 10, then 101 pairs $\{(1, 11), (2, 12), \ldots, (101, 111)\}$ are matched. The frame gap is chosen from the set $\{1, 10 : 10 : 90\}$. For each frame gap, we evaluate the methods using the mean and std of three metrics: the final objective value (larger is better), matching accuracy, and runtime (refer to Fig. 6).

Comparison. Our method achieves a very similar (slightly higher) objective value as compared to FGM, a 4 percent improvement (on average) over IPFP$_U$ and IFPU$_S$, and a 17 percent improvement over the penalty method. The relative performance of different methods evaluated by accuracy is generally consistent with that of objective value, and our method and FGM outperform other methods. In terms of runtime, our method is slower than the linear relaxation, IFPU$_U$ and IFPU$_S$, but much faster than FGM and the penalty method. These comparisons demonstrate that our method achieves state-of-the-art results in the presence of application-specific methods, while being superior to other generic IP solvers.

7 Information Theoretic Clustering

7.1 Formulation

The information theoretic clustering (ITC) model proposed in [2] is originally formulated

$$\min_{\mathbf{Y}\in\{0,1\}^{N\times K}} \text{tr}(\mathbf{Y}^\top \mathbf{W}\mathbf{Y}) \ \text{ s.t. } \begin{cases} \mathbf{Y}^\top \mathbf{1}_N = \frac{N}{K}\mathbf{1}_K \\ \mathbf{Y}\mathbf{1}_K = \mathbf{1}_N, \end{cases} \tag{27}$$

where $\mathbf{Y}$ denotes the cluster membership matrix: if $\mathbf{Y}_{ij} = 1$, then the instance feature vector $\mathbf{r}_i$ is assigned to the jth cluster. N and K denote the number of instances and clusters respectively. The column constraint $\mathbf{Y}^\top \mathbf{1}_N = \frac{N}{K}\mathbf{1}_K$ encourages the clusters to have equal-size. For details on the validity of this assumption, we refer to [2]. The row constraint $\mathbf{Y}\mathbf{1}_K = \mathbf{1}_N$ enforces that each instance can only be a member of one cluster. $\mathbf{W}$ denotes the similarity matrix: $\mathbf{W}_{ij} = \log(\|\mathbf{r}_i - \mathbf{r}_j\|_2^2)$.

BQP Reformulation. Proposition 3 allows us to reformulate Eq. (27) into standard BQP form, as shown in Eq. (28). Clearly, Eq. (28) can be solved using our ℓ_p-box ADMM algorithm. Although $\overline{\mathbf{L}}$ is large in size, it has a repetitive block structure and is extremely sparse, two properties that we exploit to make the PCG implementation much more efficient. It is noteworthy to point out that it is easy to adjust our ADMM algorithm to operate on the matrix variable $\mathbf{Y}$ directly instead of vectorizing it.

Proposition 3. *The optimization problem in Eq. (27) can be equivalently reformulated into BQP form, as follows:*

3. We also used Branch-and-Bound and Branch-and-Cut using some off-the-shelf optimization toolboxes (e.g., Hybrid [52] and OPTI [53]). However, they cannot output comparable results with other methods in reasonable time (hours). As such, we did not compare with them.

TABLE 2
Clustering Results Showing Mean(std) Values of Three Measures: RI Score (%), Objective of Eq. (27), and Runtime (Seconds), Over 10 Random Runs of All Methods

dataset →	**iris** (N = 150 instances, K = 3 clusters)			**wine** (N = 178 instances, K = 3 clusters)			**glass** (N = 214 instances, K = 6 clusters)			**letter** (N = 2e4 instances, K = 26 clusters)		
method ↓	RI	objective	runtime	RI	objective	runtime	RI	objective	runtime	RI	objective	runtime
K-means	87.37(0)	−3886(0)	**4e-3(1.6e-3)**	**93.57(0.53)**	−3504(13.2)	**7e-3(4e-3)**	67.38(2.11)	−9534(145)	**1.1e-2(4e-3)**	92.95(0.08)	−1770122(12491)	**2.68(0.86)**
penalty [17]	**94.95(0)**	**−3918(0)**	77.6(1.2)	77.80(5.81)	−3171(85)	13.7(1.06)	54.74(8.5)	−9627(438)	2060(411)	91.26(2.88)	−1634508(95624)	20523(14492)
SDP [2]	**94.95(0)**	−3890(0)	380.2(5.2)	92.74(0)	**−3533(0)**	1206(96)	**71.30(0.43)**	−9624(222)	3419.8(3.2)	N/A	N/A	N/A
$\ell_{0.5}$-box ADMM	**94.95(0)**	**−3918(0)**	0.58(0.08)	92.74(0)	**−3533(0)**	0.46(0.09)	62.21(0.26)	**−9901(95)**	0.66(0.29)	94.35(0)	−1996916(2613)	142(0.62)
ℓ_1-box ADMM	**94.95(0)**	**−3918(0)**	0.38(0.02)	92.74(0)	**−3533(0)**	0.48(0.02)	64.08(0.49)	−9841(136)	0.73(0.06)	**94.39(0)**	**−2004188(1009)**	54(17.1)
ℓ_2-box ADMM	**94.95(0)**	**−3918(0)**	0.18(0.01)	92.74(0)	**−3533(0)**	0.84(0.01)	61.39(0.13)	**−9901(59)**	0.45(0.26)	93.80(0.05)	−2002874(3015)	81(37.3)
ℓ_5-box ADMM	**94.95(0)**	**−3918(0)**	0.16(0.01)	92.74(0)	**−3533(0)**	0.46(0.02)	62.72(0.07)	−9889(108)	0.61(0.22)	93.70(0.15)	−2001653(2401)	153(1.7)
ℓ_{10}-box ADMM	**94.95(0)**	**−3918(0)**	0.15(0.01)	92.74(0)	**−3533(0)**	0.30(0.01)	64.08(0.40)	−9841(80)	1.378(0.35)	93.28(0.20)	−1992118(6934)	157(2.1)

The best value in each column is highlighted in bold.

$$\min_{\mathbf{y}\in\{0,1\}^n} \quad \mathbf{y}^\top \overline{\mathbf{L}}\mathbf{y} \quad \text{s.t.} \quad \mathbf{C}_1\mathbf{y} = \mathbf{d}_1, \tag{28}$$

where $\mathbf{y} = \text{vec}(\mathbf{Y})$ *and* $n = NK$. *The positive semi-definite matrix* $\overline{\mathbf{L}} = \mathbf{I}_K \otimes \mathbf{L} \in \mathbb{R}^{n\times n}$, *where* $\mathbf{L} = \mathbf{D} + \mathbf{W}$, *and* $\mathbf{D} = \text{diag}(\mathbf{d})$ *with* $d_i = -\sum_j^N \mathbf{W}_{ij}$. $\mathbf{C}_1 = [\mathbf{I}_K \otimes \mathbf{1}_N^\top; \mathbf{1}_K^\top \otimes \mathbf{I}_N] \in \{0,1\}^{(N+K)\times n}$, $\mathbf{d}_1 = [N/K\mathbf{1}_K; \mathbf{1}_N]$.

Proof. As the entries of $\mathbf{W}$ in (27) are non-positive, we first conduct the first transformation,

$$\begin{aligned} \text{tr}(\mathbf{Y}^\top\mathbf{W}\mathbf{Y}) &= \text{tr}(\mathbf{Y}^\top(-\mathbf{M})\mathbf{Y}) \\ &= \text{tr}(\mathbf{Y}^\top\mathbf{L}\mathbf{Y}) - \text{tr}(\mathbf{Y}^\top\mathbf{D}\mathbf{Y}) \\ &= \text{tr}(\mathbf{Y}^\top\mathbf{L}\mathbf{Y}) + const, \end{aligned} \tag{29}$$

where $\mathbf{M} = -\mathbf{W} = \mathbf{D} - \mathbf{L}$ can be seen as a similarity matrix with non-negative entries, and the diagonal matrix $\mathbf{D}$ and the positive semi-definite matrix $\mathbf{L}$ indicate the corresponding degree matrix and un-normalized Laplacian matrix respectively. During the derivation of (29), we utilize the following fact

$$\begin{aligned} \text{tr}(\mathbf{Y}^\top\mathbf{D}\mathbf{Y}) &= \sum_j^K \sum_i^N d_i \times \mathbf{Y}_{ij}^2 = \sum_j^K \sum_i^N d_i \times \mathbf{Y}_{ij} \\ &= \text{tr}[(\mathbf{d}\mathbf{1}_K^\top)^\top\mathbf{Y}] = \text{tr}[\mathbf{d}^\top(\mathbf{Y}\mathbf{1}_K)] \\ &= \text{tr}(\mathbf{d}^\top\mathbf{1}_N) = \text{const}, \end{aligned} \tag{30}$$

where the constraints $\mathbf{Y}_{ij} \in \{0,1\}$ imply that $\mathbf{Y}_{ij} = \mathbf{Y}_{ij}^2$. By vectorizing matrix $\mathbf{Y}$, we obtain the problem in Eq. (28). Since both the un-normalized Laplacian matrix $\mathbf{L}$ and $\mathbf{I}_K$ are PSD matrices and due to the property of Kronecker products, it is easy to see that $\overline{\mathbf{L}}$ is also PSD. Then, Problem (28) is a standard BQP. □

Popular Methods. In [2], Eq. (27) is solved using SDP relaxation, by replacing $\mathbf{Y}$ with $\mathbf{G} = \mathbf{Y}\mathbf{Y}^\top$. The binary constraints are also substituted with $\mathbf{G}_{ij} \in [0,1]$, $\mathbf{G}_{ii} = 1$ and $\mathbf{G} \succeq 0$. Any off-the-shelf SDP solver can be used to optimize this relaxed problem and a randomized algorithm is adopted to recover back the original variable $\mathbf{Y}$. Note that this algorithm cannot guarantee a feasible binary solution. Since $N \gg K$ in general, optimizing $\mathbf{G}$ using SDP relaxation is much more expensive than directly optimizing $\mathbf{Y}$. We will validate this in our experiments. Interestingly, LP relaxation will lead to a trivial non-binary clustering solution $\mathbf{Y} = \frac{1}{K}\mathbf{1}_{N\times K}$, at which the objective is zero. This arises because the Laplacian matrix $\mathbf{L}$ has a zero eigenvalue corresponding to the eigenvector $\mathbf{1}$. Consequently, BB and CP based on LP relaxation also fail to give good results. Thus, we do not compare against them in this application.

7.2 Clustering Experiments on UCI Data

We test the ITC model (see Eq. (27)) on four benchmark UCI datasets [55], including *iris*, *wine*, *glass* and *letter*.

Experimental Setup. We compare our method against K-means, penalty method [17], and SDP relaxation (used in [2]) that is implemented by the CVX toolbox. Note that all methods except K-means optimize Eq. (27). K-means is not only considered as a baseline, but also used as the initialization of penalty and our method. Three metrics are adopted, including the final objective value of Eq. (27) (lower is better), Rand Index (RI) and runtime. Each method is run 10 times with random (K-means) initializations and the mean and std values of these metrics are reported.

Comparison. Clustering results are summarized in Table 2. Note that we do not report the results of SDP on the *letter* dataset because it could not converge in a reasonable amount of time. Our method with different p values present similar results in all datasets. Our method improves (in objective value) over the penalty method by $[0, 11.42, 2.85, 22.62]\%$ on the four datasets respectively. It outperforms SDP relaxation by $[0.72, 0, 2.88]\%$ on the small scale datasets. It is notable that on *iris* and *letter*, the objective values and the RI values are consistent, i.e., the lower objective value corresponds to the higher RI value. In contrast, these two values are inconsistent on *wine* and *glass*. The reason is that the ITC model enforces the sizes of all clusters to be equal through the constraint $\mathbf{Y}^\top\mathbf{1}_N = \frac{N}{K}\mathbf{1}_K$ in (27). Hence, if the ground-truth cluster distribution is uneven, then the lower objective value of ITC may not indicate the better performance. The cluster distributions of *iris* and *letter* are even, while the distribution of *wine* is uneven (i.e., (59, 71, 48)), and that of *glass* is the most uneven (i.e., (70, 76, 17, 13, 9, 29)). This explains that the objective value and the RI value are a bit inconsistent on *wine*, while they are significantly inconsistent on *glass*. Overall, our method shows much better performance (i.e., the lowest objective value) on this task than other generic IP methods.

8 Conclusions and Future Work

In this work, we propose a generic IP framework called ℓ_p-box ADMM, which harnesses the attractive properties of

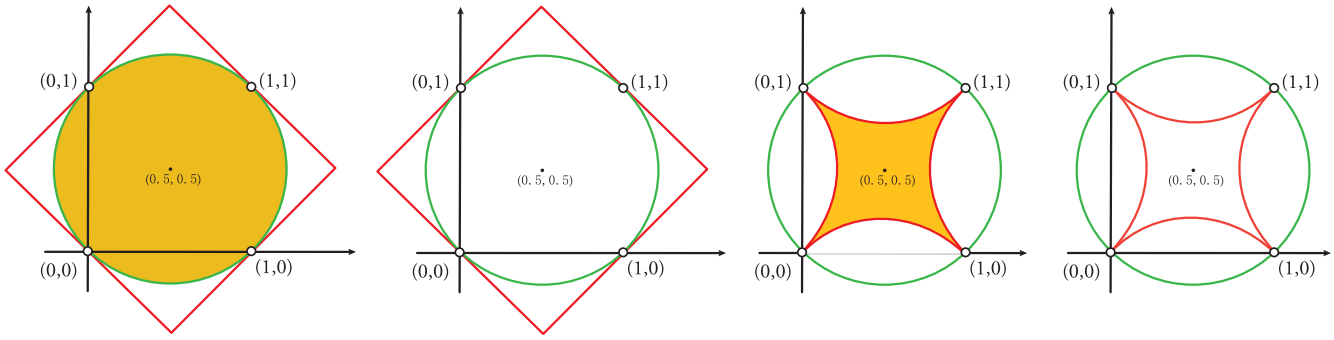

Fig. 7. Four examples of generating the set of binary vectors in $\mathbb{R}^2$ by intersecting of a pair of continuous sets using different ℓ_p norm spheres and balls. For example, the first plot on the left is the intersection of a shifted ℓ_2-ball $\{\mathbf{x} : \|\mathbf{x} - \frac{1}{2}\|_2^2 \leq \frac{1}{2}\}$ and the shifted ℓ_1-sphere $\{\mathbf{x} : \|\mathbf{x} - \frac{1}{2}\|_1 = 1\}$.

ADMM in the continuous domain by replacing the discrete constraints with equivalent and simple continuous constraints. When applied to a popular sub-class of IP problems (namely quadratic problems), our method leads to simple, computationally efficient, and provably convergent update steps. Our experiments on MRF energy minimization, graph matching, and clustering verify the superiority of our method in terms of accuracy and runtime, especially when compared to other generic IP solvers. There are many avenues of improving this framework further and we call on the community to pursue them with us.

Algorithmic Improvements. The performance/runtime of our method can be further improved in a number of ways, namely by adopting popular variants of the general ADMM framework. This includes efficient parallel and distributed computing by invoking the separability property of ADMM, where smaller pieces of the variable $\mathbf{x}$ can be updated in parallel. Moreover, linearized and stochastic variants of ADMM can be employed to approximate and therefore expedite the variable updates, especially the $\mathbf{x}$-update step. Of course, our proposed method can benefit from hardware acceleration, e.g., using GPU accelerated CG solvers for large sparse linear systems. Also, we will investigate an adaptive and systematic strategy to set γ and ρ per iteration to allow for faster runtime. Finally, we will study other popular types of $f(\mathbf{x})$ (e.g., total variation) and $\mathcal{C}$ (e.g., quadratic constraints) as specific instances of the general framework to handle other important discrete problems. For these families of problems and similar to our treatment of BQPs, we will propose efficient ways to perform the ADMM update steps. In the *supplementary material*, available online, we show that the proposed ℓ_p-box ADMM can easily handle the non-smooth objective function with ℓ_l regularization, which is popular in image denoising problems that use total variation (TV) regularization [56].

Theoretical Improvements. We will extend our framework to handle other equivalences to the $\{0,1\}^n$ binary space (refer to Fig. 7 for some examples). Ideally, we would like to propose a systematic strategy that can recommend equivalences that might be best suited for different forms of objectives and constraints.

More Applications. Because our method does not add any restrictions to the objective, our method can ideally be applied to any discrete optimization problem, such as subset selection [57], active batch selection [58], multi-label learning [59], [60], [61], [62], hash code learning [63], tracking [64], TV-based image denoising [56], to name a few. Besides, the more general mixed-integer program (MIP) problem can also be solved using our method, through equivalently replacing the integer constraints by continuous constraints. They will be explored in future work. The codes of this work are available at: https://github.com/wubaoyuan/Lpbox-ADMM.

Acknowledgments

This work is supported by competitive research funding from King Abdullah University of Science and Technology (KAUST), and Tencent AI Lab. The authors would like to thank Prof. Siwei Lyu, Prof. Wotao Yin, and Dr. Li Shen for their constructive comments and discussions for this work. Baoyuan Wu and Bernard Ghanem contributed equally to this work.

References

[1] S. Ramalingam, P. Kohli, K. Alahari, and P. H. Torr, "Exact inference in multi-label CRFs with higher order cliques," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8.
[2] M. Wang and F. Sha, "Information theoretical clustering via semidefinite programming," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 761–769.
[3] P. Bertolazzi, G. Felici, P. Festa, G. Fiscon, and E. Weitschek, "Integer programming models for feature selection: New extensions and a randomized solution algorithm," *Eur. J. Oper. Res.*, vol. 250, pp. 389–399, 2016.
[4] A. Joulin, F. Bach, and J. Ponce, "Discriminative clustering for image co-segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1943–1950.
[5] P. Wang, C. Shen, A. V. D. Hengel, and P. H. S. Torr, "Large-scale binary quadratic optimization using semidefinite relaxation and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 3, pp. 470–485, 2017.
[6] X. Mei, H. Qi, B.-G. Hu, and S. Lyu, "Improving image restoration with soft-rounding," in *Proc. IEEE Conf. Comput. Vis.*, 2015, pp. 459–467.
[7] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2009, pp. 1753–1760.
[8] M. Leordeanu, M. Hebert, and R. Sukthankar, "An integer projected fixed point method for graph matching and map inference," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2009, pp. 1114–1122.
[9] F. Zhou and F. De la Torre, "Factorized graph matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 127–134.
[10] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for Markov random fields," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 16–29.
[11] L. Hong and G. Chen, "Segment-based stereo matching using graph cuts," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2004, pp. I-74–I-81.
[12] S. Z. Li, *Markov Random Field Modeling in Image Analysis*. Berlin, Germany: Springer, 2009.

[13] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr, "Interactive image segmentation using an adaptive GMMRF model," in *Proc. Eur. Conf. Comput. Vis.*, 2004, pp. 428–441.

[14] C. Rother, V. Kolmogorov, and A. Blake, "GrabCut: Interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, 2004.

[15] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica: J. Econometric Soc.*, vol. 28, pp. 497–520, 1960.

[16] J. Kelley, "The cutting-plane method for solving convex programs," *J. Soc. Ind. Appl. Math.*, vol. 8, pp. 703–712, 1960.

[17] W. Murray and K.-M. Ng, "An algorithm for nonlinear optimization problems with binary variables," *Comput. Optimization Appl.*, vol. 47, no. 2, pp. 257–288, 2010.

[18] S. Lucidi and F. Rinaldi, "Exact penalty functions for nonlinear integer programming problems," *J. Optimization Theory Appl.*, vol. 145, no. 3, pp. 479–488, 2010.

[19] M. De Santis, "Continuous approaches to mixed integer programming problems," PhD Thesis, Sapienza University of Rome, 2012.

[20] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[21] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 9, pp. 1124–1137, Sep. 2004.

[22] J. Munkres, "Algorithms for the assignment and transportation problems," *J. Soc. Ind. Appl. Math.*, vol. 5, no. 1, pp. 32–38, 1957.

[23] L. A. Wolsey and G. L. Nemhauser, *Integer and Combinatorial Optimization*. Hoboken, NJ, USA: Wiley, 2014.

[24] M. J. Wainwright, M. I. Jordan, et al., "Graphical models, exponential families, and variational inference," *Found. Trends Mach. Learn.*, vol. 1, no. 1/2, pp. 1–305, 2008.

[25] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[26] S. Mehrotra, "On the implementation of a primal-dual interior point method," *SIAM J. Optimization*, vol. 2, no. 4, pp. 575–601, 1992.

[27] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[28] M. Laurent, "A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0-1 programming," *Math. Operations Res.*, vol. 28, no. 3, pp. 470–496, 2003.

[29] J. B. Lasserre, "An explicit exact SDP relaxation for nonlinear 0-1 programs," in *Integer Programming and Combinatorial Optimization*. Berlin, Germany: Springer, 2001, pp. 293–303.

[30] M. Laurent and F. Rendl, *Semidefinite Programming and Integer Programming*. Amsterdam, The Netherlands: Centrum voor Wiskunde en Informatica, 2002.

[31] P. Wang, C. Shen, and A. van den Hengel, "A fast semidefinite approach to solving binary quadratic problems," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 1312–1319.

[32] N. Parikh and S. Boyd, "Block splitting for distributed optimization," *Math. Program. Comput.*, vol. 6, no. 1, pp. 77–102, 2014.

[33] P. Ravikumar and J. Lafferty, "Quadratic programming relaxations for metric labeling and Markov random field map estimation," in *Proc. Int. Conf. Mach. Learn.*, 2006, pp. 737–744.

[34] S. Kim and M. Kojima, "Second order cone programming relaxation of nonconvex quadratic optimization problems," *Optimization Methods Softw.*, vol. 15, no. 3/4, pp. 201–224, 2001.

[35] S. Burer, "On the copositive representation of binary and continuous nonconvex quadratic programs," *Math. Program.*, vol. 120, no. 2, pp. 479–495, 2009.

[36] C. Yu, K. L. Teo, and Y. Bai, "An exact penalty function method for nonlinear mixed discrete programming problems," *Optimization Lett.*, vol. 7, no. 1, pp. 23–38, 2013.

[37] Y. Xu, W. Yin, Z. Wen, and Y. Zhang, "An alternating direction algorithm for matrix completion with nonnegative factors," *Frontiers Math. China*, vol. 7, no. 2, pp. 365–384, 2012.

[38] B. Jiang, S. Ma, and S. Zhang, "Alternating direction method of multipliers for real and complex polynomial optimization models," *Optimization*, vol. 63, no. 6, pp. 883–898, 2014.

[39] L. Yang, T. K. Pong, and X. Chen, "Alternating direction method of multipliers for nonconvex background/foreground extraction," *SIAM J. Imaging Sci.*, vol. 10, no. 1, pp. 74–110, 2017.

[40] G. Li and T. K. Pong, "Global convergence of splitting methods for nonconvex composite optimization," *SIAM J. Optimization*, vol. 25, no. 4, pp. 2434–2460, 2015.

[41] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," *J. Sci. Comput.*, pp. 1–35, 2015.

[42] W. Wang and M. A. Carreira-Perpinán, "Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application," arXiv:1309.1541, 2013.

[43] M. Coste, "An introduction to semialgebraic geometry," *RAAG Netw. School*, vol. 145, 2002, Art. no. 30.

[44] Q. Fu, H. Wang, and A. Banerjee, "Bethe-ADMM for tree decomposition based parallel map inference," in *Proc. Conf. Uncertainty Artif. Intell.*, 2013, pp. 222–231.

[45] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA, USA: MIT Press, 2009.

[46] F. Bach, "Learning with submodular functions: A convex optimization perspective," *Found. Trends Mach. Learn.*, vol. 6, no. 2-3, pp. 145–373, 2013.

[47] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov, "Fast approximate energy minimization with label costs," *Int. J. Comput. Vis.*, vol. 96, no. 1, pp. 1–27, 2012.

[48] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr, "Interactive image segmentation using an adaptive GMMRF model," in *Proc. Eur. Conf. Comput. Vis.*, 2004, pp. 428–441.

[49] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller, "Multi-class segmentation with relative location prior," *Int. J. Comput. Vis.*, vol. 80, no. 3, pp. 300–316, 2008.

[50] M. Leordeanu, R. Sukthankar, and M. Hebert, "Unsupervised learning for graph matching," *Int. J. Comput. Vis.*, vol. 96, no. 1, pp. 28–45, 2012.

[51] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Res. Logistics Quart.*, vol. 3, no. 1/2, pp. 95–110, 1956.

[52] A. Bemporad, "Hybrid toolbox - user's guide," 2004. [Online]. Available: http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox

[53] J. Currie and D. I. Wilson, "OPTI: Lowering the barrier between open source optimizers and the industrial MATLAB user," in *Foundations of Computer-Aided Process Operations*, N. Sahinidis and J. Pinto, Eds. Savannah, GA, USA, 2012.

[54] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *Proc. IEEE Conf. Comput. Vis.*, 2005, pp. 1482–1489.

[55] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[56] A. Chambolle, "Total variation minimization and a class of binary MRF models," in *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Berlin, Germany: Springer, 2005, pp. 136–152.

[57] C. Qian, Y. Yu, and Z.-H. Zhou, "Subset selection by Pareto optimization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 1765–1773.

[58] S. Chakraborty, V. Balasubramanian, Q. Sun, S. Panchanathan, and J. Ye, "Active batch selection via convex relaxations with guaranteed solution bounds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 10, pp. 1945–1958, Oct. 2015.

[59] B. Wu, Z. Liu, S. Wang, B.-G. Hu, and Q. Ji, "Multi-label learning with missing labels," in *Proc. Int. Conf. Pattern Recognit.*, 2014, pp. 1964–1968.

[60] B. Wu, S. Lyu, and B. Ghanem, "ML-MG: Multi-label learning with missing labels using a mixed graph," in *Proc. IEEE Conf. Comput. Vis.*, 2015, pp. 4157–4165.

[61] M.-L. Zhang and L. Wu, "Lift: Multi-label learning with label-specific features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 1, pp. 107–120, Jan. 2015.

[62] B. Wu, S. Lyu, and B. Ghanem, "Constrained submodular minimization for missing labels and class imbalance in multi-label learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 2229–2236.

[63] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised discrete hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 37–45.

[64] B. Wang, G. Wang, K. L. Chan, and L. Wang, "Tracklet association by online target-specific metric learning and coherent dynamics estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 3, pp. 589–602, Mar. 2017.

Baoyuan Wu is currently a senior research scientist with Tencent AI Lab. He received the PhD degree from the National Laboratory of Pattern Recognition, Chinese Academy of Sciences (CASIA), in 2014, supervised by Prof. Baogang Hu and Prof. Qiang Ji. He was a Postdoc with IVUL lab, King Abdullah University of Science and Technology (KAUST), working with Prof. Bernard Ghanem, from August 2014 to November 2016. His research interests include machine learning and computer vision, including probabilistic graphical models, structured output learning, multi-label learning, and integer programming. His work has been published in the *International Journal of Computer Vision*, CVPR, ICCV, ECCV, AAAI, etc. He is a member of the IEEE.

Bernard Ghanem is currently an associate professor with the King Abdullah University of Science and Technology (KAUST), in the Visual Computing Center (VCC). He leads the Image and Video Understanding Lab (IVUL), KAUST. He is involved in several interesting projects that focus on exploiting techniques in computer vision and machine learning for real-world applications including semantic sports video analysis, large-scale activity recognition/detection, and real-time crowd analysis. He has published more than 100 peer-papers in peer-reviewed venues including the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, the *International Journal of Computer Vision*, CVPR, ICCV, ECCV, etc. He is a member of the IEEE.