Optimization Methods & Software, 2017
http://dx.doi.org/10.1080/10556788.2017.1304548

A general system for heuristic minimization of convex functions over non-convex sets

S. Diamond*, R. Takapoui and S. Boyd

Departments of CS and EE, Stanford University, Stanford, CA, USA

(*Received 17 May 2016; accepted 6 March 2017*)

We describe general heuristics to approximately solve a wide variety of problems with convex objective and decision variables from a non-convex set. The heuristics, which employ convex relaxations, convex restrictions, local neighbour search methods, and the alternating direction method of multipliers, require the solution of a modest number of convex problems, and are meant to apply to general problems, without much tuning. We describe an implementation of these methods in a package called NCVX, as an extension of CVXPY, a Python package for formulating and solving convex optimization problems. We study several examples of well known non-convex problems, and show that our general purpose heuristics are effective in finding approximate solutions to a wide variety of problems.

Keywords: non-convex optimization; convex approximations; heuristics; alternating direction method of multipliers; modelling software

AMS Subject Classification: 90C59; 90C25; 90C26

1. Introduction

1.1 *The problem*

We consider the optimization problem

$$\begin{array}{ll} \text{minimize} & f_0(x,z) \\ \text{subject to} & f_i(x,z) \leq 0, \quad i = 1,\ldots,m \\ & Ax + Bz = c \\ & z \in \mathcal{Z}, \end{array} \tag{1}$$

where $x \in \mathbf{R}^n$ and $z \in \mathbf{R}^q$ are the decision variables, $A \in \mathbf{R}^{p\times n}$, $B \in \mathbf{R}^{p\times q}$, $c \in \mathbf{R}^p$ are problem data, and $\mathcal{Z} \subseteq \mathbf{R}^q$ is closed. We assume that the objective and inequality constraint functions $f_0,\ldots,f_m : \mathbf{R}^n \times \mathbf{R}^q \to \mathbf{R}$ are jointly convex in x and z. When the set $\mathcal{Z}$ is convex, (1) is a convex optimization problem, but we are interested here in the case where $\mathcal{Z}$ is not convex. Roughly speaking, the problem (1) is a convex optimization problem, with some additional non-convex constraints, $z \in \mathcal{Z}$. We can think of x as the collection of decision variables that appear only in convex constraints, and z as the decision variables that are directly constrained to lie in

*Corresponding author. Email: diamond@cs.stanford.edu the (generally) non-convex set $\mathcal{Z}$. The set $\mathcal{Z}$ is often a Cartesian product, $\mathcal{Z} = \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_k$, where $\mathcal{Z}_i \subset \mathbf{R}^{q_i}$ are sets that are simple to describe, for example, $\mathcal{Z}_i = \{0, 1\}$. We denote the optimal value of the problem (1) as $p^\star$, with the usual conventions that $p^\star = +\infty$ if the problem is infeasible, and $p^\star = -\infty$ if the problem is unbounded below.

1.2 *Special cases*

Mixed integer convex optimization. When $\mathcal{Z} = \{0, 1\}^q$, the problem (1) is a general mixed integer convex program, that is, a convex optimization problem in which some variables are constrained to be Boolean. ('Mixed Boolean' would be a more accurate name for such a problem, but 'mixed integer' is commonly used.) It follows that the problem (1) is NP-hard; it includes as a special case, for example, the general Boolean satisfaction problem.

Cardinality constrained convex optimization. As another broad special case of (1), consider the case $\mathcal{Z} = \{z \in \mathbf{R}^q \mid \mathbf{card}(z) \le k,\ \|z\|_\infty \le M\}$, where $\mathbf{card}(z)$ is the number of non-zero elements of z, and k and M are given. We call this the general *cardinality-constrained convex problem*. It arises in many interesting applications, such as regressor selection.

Other special cases. As we will see in Section 6, many (hard) problems can be formulated in the form (1). More examples include regressor selection, 3-SAT, circle packing, the travelling salesman problem (TSP), factor analysis modelling, inexact graph isomorphism, and many more.

1.3 *Convex relaxation*

Convex relaxation of a set. For bounded sets $\mathcal{Z}$ there usually is a manageable full or partial description of the convex hull of $\mathcal{Z}$. By this we mean a (modest-sized) set of convex inequality and linear equality constraints that hold for every $z \in \mathcal{Z}$:

$$z \in \mathcal{Z} \implies h_i(z) \le 0, \quad i = 1, \ldots, s, \quad Fz = g.$$

We will assume that these relaxation constraints are included in the convex constraints of (1). Adding these relaxation constraints to the original problem yields an equivalent problem (since the added constraints are redundant), but can improve the convergence of any method, global or heuristic. By tractable, we mean that the number of added constraints is modest, and in particular, polynomial in q.

For example, when $\mathcal{Z} = \{0, 1\}^q$, we have the inequalities $0 \le z_i \le 1$, $i = 1, \ldots, q$. (These inequalities define the convex hull of $\mathcal{Z}$, that is, all other convex inequalities that hold for all $z \in \mathcal{Z}$ are implied by them.) When

$$\mathcal{Z} = \{z \in \mathbf{R}^q \mid \mathbf{card}(z) \le k,\ \|z\|_\infty \le M\},$$

we have the convex inequalities

$$\|z\|_1 \le kM, \quad \|z\|_\infty \le M.$$

(These inequalities define the convex hull of $\mathcal{Z}$.) For general bounded $\mathcal{Z}$ the inequality $\|z\|_\infty \le M$ will always be a convex relaxation for some value of M.

Relaxed problem. If we remove the non-convex constraint $z \in \mathcal{Z}$, we get a *convex relaxation* of the original problem:

$$\begin{array}{ll} \text{minimize} & f_0(x,z) \\ \text{subject to} & f_i(x,z) \leq 0, \quad i = 1,\dots,m \\ & Ax + Bz = c. \end{array} \tag{2}$$

(Recall that convex equalities and inequalities known to hold for $z \in \mathcal{Z}$ have been incorporated in the convex constraints.) The relaxed problem is convex; its optimal value is a lower bound on the optimal value $p^\star$ of (1). A solution (x^*, z^*) to problem (2) need not satisfy $z^* \in \mathcal{Z}$, but if it does, the pair (x^*, z^*) is optimal for (1).

1.4 *Projections and approximate projections*

Our methods will make use of tractable projection, or tractable approximate projection, onto the set $\mathcal{Z}$. The usual Euclidean projection onto $\mathcal{Z}$ will be denoted Π. (It need not be unique when $\mathcal{Z}$ is not convex.) By approximate projection, we mean any function $\hat{\Pi} : \mathbf{R}^q \to \mathcal{Z}$ that satisfies $\hat{\Pi}(z) = z$ for $z \in \mathcal{Z}$. A useful approximate projection $\hat{\Pi}(z)$ will also approximately minimize $\|u - z\|_2^2$ over $u \in \mathcal{Z}$, but since all the algorithms we present are heuristics, we do not formalize this requirement. We use approximate projections when computing an exact projection onto the set $\mathcal{Z}$ is too expensive.

For example, when $\mathcal{Z} = \{0,1\}^q$, exact projection is given by rounding the entries to $\{0,1\}$. As a less trivial example, consider the cardinality-constrained problem. The projection of z onto $\mathcal{Z}$ is given by

$$(\Pi(z))_i = \begin{cases} M & z_i > M,\ i \in \mathcal{I}, \\ -M & z_i < -M,\ i \in \mathcal{I}, \\ z_i & |z_i| \leq M,\ i \in \mathcal{I}, \\ 0 & i \notin \mathcal{I}, \end{cases}$$

where $\mathcal{I} \subseteq \{1,\dots,q\}$ is a set of indices of k largest values of $|z_i|$. We will describe many projections, and some approximate projections, in Section 4.

1.5 *Residual and merit functions*

For any (x,z) with $z \in \mathcal{Z}$, we define the *constraint residual* as

$$r(x,z) = \sum_{i=1}^{m} (f_i(x,z))_+ + \|Ax + Bz - c\|_1,$$

where $(u)_+ = \max\{u, 0\}$ denotes the positive part; (x,z) is feasible if and only if $r(x,z) = 0$. Note that $r(x,z)$ is a convex function of (x,z). We define the *merit function* of a pair (x,z) as

$$\eta(x,z) = f_0(x,z) + \lambda r(x,z),$$

where $\lambda > 0$ is a parameter. The merit function is also a convex function of (x,z).

When $\mathcal{Z}$ is convex and the problem is feasible, minimizing $\eta(x,z)$ for large enough λ yields a solution of the original problem (1) (that is, the residual is a so-called exact penalty function); when the problem is not feasible, it tends to find approximate solutions that satisfy many of the constraints [23,27,38].

We will use the merit function to judge candidate approximate solutions (x, z) with $z \in \mathcal{Z}$; that is, we take a pair with lower merit function value to be a better approximate solution than one with higher merit function value. For some problems (for example, unconstrained problems) it is easy to find feasible points, so all candidate points will be feasible. The merit function then reduces to the objective value. At the other extreme, for feasibility problems the objective is zero, and the goal is to find a feasible point. In this case the merit function reduces to $\lambda r(x, z)$, that is, a positive multiple of the residual function.

1.6 *Solution methods*

In this section we describe various methods for solving the problem (1), either exactly (globally) or approximately.

Global methods. Depending on the set $\mathcal{Z}$, the problem (1) can be solved globally by a variety of algorithms, including (or mixing) branch-and-bound [10,54,62], branch-and-cut [64,80,82], semidefinite hierarchies [76], or even direct enumeration when $\mathcal{Z}$ is a finite set. In each iteration of these methods, a convex optimization problem derived from (1) is solved, with $\mathcal{Z}$ removed, and (possibly) additional variables and convex constraints added. While for many applications these methods are effective, they are generally thought to have high worst-case complexities and indeed can be very slow for some problems.

Local solution methods and heuristics. A local method for (1) solves a modest number of convex problems, in an attempt to find a good approximate solution, that is, a pair (x, z) with $z \in \mathcal{Z}$ and a low value of the merit function $\eta(x, z)$. For a feasibility problem, we might hope to find a solution; and if not, find one with a small constraint residual. For a general problem, we can hope to find a feasible point with low objective value, ideally near the lower bound on $p^\star$ from the relaxed problem. If we cannot find any feasible points, we can settle for a pair (x, z) with $z \in \mathcal{Z}$ and low merit function value. All of these methods are heuristics, in the sense that they cannot in general be guaranteed to find an optimal, or even good, or even feasible, point in only a modest number of iterations.

There are of course many heuristics for the general problem (1) and for many of its special cases. For example, any global optimization method can be stopped after some modest number of iterations; we then take the best point found (in terms of the merit function) as our approximate solution. (We will discuss some local search methods, including neighbour search and polishing, in Section 2.)

1.7 *Our approach*

The purpose of this paper is to describe a general system for heuristic solution of (1), based on solving a *modest* number of convex problems derived from (1). By heuristic, we mean that the algorithm need not find an optimal point, or indeed, even a feasible point, even when one exists. We would hope that for many feasible problem instances from some application, the algorithm does find a feasible point, and one with objective not too far from the optimal value. The disadvantage of a heuristic over a global method is clear and simple: it need not find an optimal point. The advantage of a heuristic is that it can be (and often is) dramatically faster to carry out than a global method. Moreover there are many applications where a heuristic method for (1) is sufficient because the difference between a globally optimal solution and a solution that is only close to optimal is not significant in practice.

ADMM. One of the heuristic methods described in this paper is based on the alternating directions method of multipliers (ADMM), an operator splitting algorithm originally devised to solve convex optimization problems (see, e.g. [8,25] for comprehensive tutorials on ADMM). We call this heuristic non-convex alternating directions method of multipliers (NC-ADMM). ADMM was introduced in the mid-1970s [30,35]. More recently, ADMM has found applications in a variety of distributed settings in machine learning such as model fitting, resource allocation, and classification (see, e.g. [4,61,72,74,88,89,92]). The idea of using ADMM as a general purpose heuristic to solve non-convex problems was mentioned in [8, Ch. 9] and was further explored in [20]. Consensus ADMM has been used for non-convex quadratically constrained quadratic programs in [48]. In [91], ADMM has been applied to non-negative matrix factorization with missing values. ADMM also has been used for real and complex polynomial optimization models in [49], for constrained tensor factorization in [57], and for optimal power flow in [26]: all non-convex problems. ADMM can be viewed as a version of the method of multipliers [5,40,68], where a Gauss–Seidel pass over x and z is used instead of the usual joint minimization. There is a long history of using the method of multipliers to (attempt to) solve non-convex problems [12,13,44,46,56,66,90]. Instead of basing our heuristic on ADMM, which is Douglas–Rachford splitting [24] applied to the dual problem, we could also have used Spingarn's method [78], which is Douglas–Rachford splitting applied directly to the primal problem. For non-convex problems the two approaches could yield different results.

Our contribution. Our main contribution is to identify a small number of concepts and methods for heuristics for non-convex problems that can be applied across a very wide variety of problems. The only essential one is a projection, or even just an approximate projection, onto the non-convex sets that appear in the problem. The others, which can dramatically improve the performance of the heuristic, are to identify a convex relaxation for each non-convex set, a convex restriction at a general point in each non-convex set, and a method to identify or list neighbours of a given point (in a discrete non-convex set). We have implemented a general purpose system that uses just these four methods and handles a variety of different problems. Our implementation is readily extensible; the user only needs to implement these four methods for any new non-convex set to be added.

Outline. The paper has the following structure. In Section 2 we discuss local search methods and describe how they can be used as solution improvement methods. This will enable us to study simple but sophisticated methods such as relax–round–polish and iterative neighbour search. In Section 3 we present a heuristic for problem (1) based on ADMM, which makes use of the solution improvement methods in Section 2. In Section 4 we catalog a variety of non-convex sets for which Euclidean projection or approximate projection is easily evaluated and, when applicable, we discuss relaxations, restrictions, and distance functions that define the set of neighbours for a given point. In Section 5 we discuss an implementation of our general system for heuristic solution NCVX, as an extension of CVXPY [22], a Python package for formulating and solving convex optimization problems. The object-oriented features of CVXPY make the extension particularly simple to implement. Finally, in Section 6 we demonstrate the performance of our methods on several example problems.

2. Local improvement methods

In this section we describe some simple general local search methods. These methods take a point $z \in \mathcal{Z}$ and by performing a local search on z they find a candidate pair $(\hat{x}, \hat{z})$, with $\hat{z} \in \mathcal{Z}$ and a lower merit function. We will see that for many applications using these methods with a good initialization will result in an approximate solution. We will also see how we can use these methods to improve solution candidates from other heuristics, hence we refer to these methods as *solution improvement*.

2.1 *Polishing*

Convex restriction. For non-discrete $\mathcal{Z}$, the idea of a *convex restriction* at a point is useful for local search methods. A convex restriction at a point $z \in \mathcal{Z}$ is a convex set $\mathcal{Z}^{\text{rstr}}(z)$ that satisfies $z \in \mathcal{Z}^{\text{rstr}}(z) \subseteq \mathcal{Z}$. The trivial restriction given by $\mathcal{Z}^{\text{rstr}}(z) = \{z\}$ is valid for all $\mathcal{Z}$. When $\mathcal{Z}$ is discrete, for example $\mathcal{Z} = \{0,1\}^q$, the trivial restriction is the only restriction. In other cases we can have interesting non-trivial restrictions. For example, with $\mathcal{Z} = \{z \in \mathbf{R}^q \mid \mathrm{card}(z) \leq k,\ \|z\|_\infty \leq M\}$, we can take as restriction $\mathcal{Z}^{\text{rstr}}(\tilde{z})$ the set of vectors z with the same sparsity pattern as $\tilde{z}$, and $\|z\|_\infty \leq M$.

Polishing. Given any point $\tilde{z} \in \mathcal{Z}$, we can replace the constraint $z \in \mathcal{Z}$ with $z \in \mathcal{Z}^{\text{rstr}}(\tilde{z})$ to get the convex problem

$$\begin{array}{ll} \text{minimize} & \eta(x,z), \\ \text{subject to} & z \in \mathcal{Z}^{\text{rstr}}(\tilde{z}), \end{array} \tag{3}$$

with variables x,z. (When the restriction $\mathcal{Z}^{\text{rstr}}(\tilde{z})$ is the trivial one, that is, a singleton, this is equivalent to fixing $z = \tilde{z}$ and minimizing over x.) We call this problem the *convex restriction* of (1) at the point $\tilde{z}$. The restricted problem is convex, and its optimal value is an upper bound on $p^\star$ assuming λ is sufficiently large in $\eta(x,z) = f_0(x,z) + \lambda r(x,z)$ to ensure $r(x,z) = 0$.

As a simple example of polishing consider the mixed integer convex problem. The only restriction is the trivial one, so the polishing problem for a given Boolean vector $\tilde{z}$ simply fixes the values of the Boolean variables and solves the convex problem over the remaining variables, that is, x. For the cardinality-constrained convex problem, polishing fixes the sparsity pattern of z and solves the resulting convex problem over z and x.

For problems with non-trivial restrictions, we can solve the polishing problem repeatedly until convergence. In other words we can use the output of the polishing problem as an initial point for another polishing problem and keep iterating until the merit function stops improving. This technique is called *iterated polishing* and described in Algorithm 1.

Algorithm 1 Iterated polishing

```
1: Input: $(\tilde{x},\tilde{z})$
2: do
3:     $(x^{\text{old}}, z^{\text{old}}) \leftarrow (\tilde{x},\tilde{z})$.
4:     Find $(\tilde{x},\tilde{z})$ by solving the polishing problem with restriction $z \in \mathcal{Z}^{\text{rstr}}(z^{\text{old}})$.
5: while $\eta(\tilde{x},\tilde{z}) < \eta(x^{\text{old}}, z^{\text{old}})$
6: return $(\tilde{x},\tilde{z})$.
```

If there exists a point $\tilde{x}$ such that $(\tilde{x},\tilde{z})$ is feasible, the restricted problem is feasible too. The restricted problem need not be feasible in general, but if it is, with solution $(\hat{x},\hat{z})$, then the pair $(\hat{x},\hat{z})$ is feasible for the original problem (1) and satisfies $f_0(\hat{x},\hat{z}) \leq f_0(\tilde{x},\tilde{z})$ for any $\tilde{x}$ for which $(\tilde{x},\tilde{z})$ is feasible. So polishing can take a point $\tilde{z} \in \mathcal{Z}$ (or a pair $(\tilde{x},\tilde{z})$) and produce another pair $(\hat{x},\hat{z})$ with a possibly better objective value.

2.2 *Relax–round–polish*

With the simple tools described so far (i.e. relaxation, polishing, and projection) we can create several heuristics for approximately solving the problem (1). A basic version solves the relaxation, projects the relaxed value of z onto $\mathcal{Z}$, and then iteratively polishes the result.

Algorithm 2 Relax–round–polish heuristic

```
1: Solve the convex relaxation (2) to obtain $(x^{\text{rlx}}, z^{\text{rlx}})$.
2: $z^{\text{rnd}} \leftarrow \Pi(z^{\text{rlx}})$.
3: Use Algorithm 1 on $(x^{\text{rlx}}, z^{\text{rnd}})$ to get $(\hat{x}, \hat{z})$.
```

Note that in the first step we also obtain a lower bound on the optimal value $p^\star$; in the polishing step we obtain an upper bound and a feasible pair $(\hat{x}, \hat{z})$ that achieves the upper bound provided that polishing is successful. The best outcome is for these bounds to be equal, which means that we have found a (global) solution of (1) (for this problem instance). But relax–round–polish can fail; for example, it can fail to find a feasible point even though one exists.

Many variations on relax–round–polish are possible. We can introduce randomization by replacing the round step with

$$z^{\text{rnd}} = \Pi(z^{\text{rlx}} + w),$$

where w is a random vector. We can repeat this heuristic with N different random instances of w. For each of N samples of w, we polish, giving us a set of N candidate approximate solutions. We then take as our final approximate solution the best among these N candidates, that is, the one with least merit function.

2.3 *Neighbour search*

Neighbours. When $\mathcal{Z}$ is discrete, convex restrictions are not useful for local search. Instead we use the concept of *neighbours* of a point $z \in \mathcal{Z}$ as a discrete analogue to a restriction. As with a restriction, we do local search over the set of neighbours. Neighbours are defined in terms of a distance function $\mathcal{Z}^{\text{dist}} : \mathcal{Z} \times \mathcal{Z} \to \mathbf{Z}_+ \cup \{+\infty\}$. The set of neighbours of a point $z \in \mathcal{Z}$ within distance $D \in \mathbf{Z}_+$, denoted $\mathcal{Z}^{\text{ngbr}}(z, D)$, is given by $\mathcal{Z}^{\text{ngbr}}(z, k) = \{\mathcal{Z}^{\text{dist}}(y, z) \leq D \mid y \in \mathcal{Z}\}$. We select a distance function and distance D such that the size of $\mathcal{Z}^{\text{ngbr}}(z, D)$ is computationally tractable for all $z \in \mathcal{Z}$. For non-discrete $\mathcal{Z}$, we use the trivial distance function

$$\mathcal{Z}^{\text{dist}}(z, y) = \begin{cases} 0 & z = y, \\ +\infty & z \neq y, \end{cases}$$

for which $\mathcal{Z}^{\text{ngbr}}(z, D) = \{z\}$ for all z and D.

For example, for the set of Boolean vectors in $\mathbf{R}^n$ we use *Hamming distance*, the number of entries in which two Boolean vectors differ. Hence the neighbours of a Boolean vector z within distance D are the set of vectors that differ from z in up to D components. We define the distance between two permutation matrices as the minimum number of swaps of adjacent rows and columns necessary to transform the first matrix into the second. With this distance metric, neighbours of a permutation matrix Z within distance D are the set of permutation matrices generated by swapping any two adjacent rows or columns in Z up to D times. We define distance in terms of swaps of adjacent rows and columns rather than swaps of arbitrary rows and columns to reduce the number of neighbours.

For Cartesian products of discrete sets we use the sum of distances. In this case, for $z = (z_1, z_2, \dots, z_q) \in \mathcal{Z} = \mathcal{Z}_1 \times \mathcal{Z}_2 \times \cdots \times \mathcal{Z}_q$, neighbours of z within distance D are points of the form $(\tilde{z}_1, \tilde{z}_2, \dots, \tilde{z}_q)$ where $\sum_{i=1}^{q} \mathcal{Z}_i^{\text{dist}}(\tilde{z}_i, z_i) \le D$.

Basic neighbour search. We introduced polishing as a tool that can find a pair $(\hat{x}, \hat{z})$ given an input $\tilde{z} \in \mathcal{Z}$ by solving a sequence of convex problems. In basic neighbour search we solve the polishing problem for $\tilde{z}$ and all neighbours of $\tilde{z}$ (within distance D) and return the pair (x^*, z^*) with the smallest merit function value. In practice, we can sample from $\mathcal{Z}^{\text{ngbr}}(\tilde{z}, D)$ instead of iterating over all points in $\mathcal{Z}^{\text{ngbr}}(\tilde{z}, D)$ if $|\mathcal{Z}^{\text{ngbr}}(\tilde{z}, D)|$ is large.

Algorithm 3 Basic neighbour search

```
1: Input: $\tilde{z}$
2: Initialize $(x_{\text{best}}, z_{\text{best}}) = \emptyset$, $\eta_{\text{best}} = \infty$.
3: for $\hat{z} \in \mathcal{Z}^{\text{ngbr}}(\tilde{z}, D)$ do
4:     Find $(x^*, z^*)$, by solving the polishing problem (3), with constraint $z \in \mathcal{Z}^{\text{rstr}}(\hat{z})$.
5:     if $\eta(x^*, z^*) < \eta_{\text{best}}$ then
6:         $(x_{\text{best}}, z_{\text{best}}) \leftarrow (x^*, z^*)$, $\eta_{\text{best}} \leftarrow \eta(x^*, z^*)$.
7:     end if
8: end for
9: return $(x_{\text{best}}, z_{\text{best}})$.
```

Iterated neighbour search. As with polishing, we can apply basic neighbour search repeatedly until convergence. In other words we can feed the output of Algorithm 3 back into Algorithm 3 until the merit function stops improving. The technique is called *iterated neighbour search* and described in Algorithm 4. Notice that for non-discrete sets where $\mathcal{Z}^{\text{ngbr}}(z, D) = \{z\}$ for all z and D, this algorithm reduces to iterated polishing.

Algorithm 4 Iterated neighbour search

```
1: Input: $(\tilde{x}, \tilde{z})$
2: do
3:     $(x^{\text{old}}, z^{\text{old}}) \leftarrow (\tilde{x}, \tilde{z})$.
4:     Use Algorithm 3 on $z^{\text{old}}$ to get $(\tilde{x}, \tilde{z})$.
5: while $\eta(\tilde{x}, \tilde{z}) < \eta(x^{\text{old}}, z^{\text{old}})$
6: return $(\tilde{x}, \tilde{z})$.
```

3. NC-ADMM

We already can use the simple tools described in the previous section as heuristics to find approximate solutions to problem (1). In this section, we describe the alternating direction method of multipliers (ADMM) as a mechanism to generate candidate points $\tilde{z}$ to carry out local search methods such as iterated neighbour search. We call this method non-convex ADMM, or NC-ADMM.

3.1 *ADMM*

Define $\phi : \mathbf{R}^q \to \mathbf{R} \cup \{-\infty, +\infty\}$ such that $\phi(z)$ is the best objective value of problem (1) after fixing z. In other words,

$$\phi(z) = \inf_x \{f_0(x,z) \mid f_i(x,z) \leq 0,\ i = 1, \ldots, m,\ Ax + Bz = c\}.$$

Notice that $\phi(z)$ can be $+\infty$ or $-\infty$ in case the problem is not feasible for this particular value of z, or problem (2) is unbounded below after fixing z. The function ϕ is convex, since it is the partial minimization of a convex function over a convex set [9, §3.4.4]. It is defined over all points $z \in \mathbf{R}^q$, but we are interested in finding its minimum value over the non-convex set $\mathcal{Z}$. In other words, problem (1) can be formulated as

$$\begin{array}{ll} \text{minimize} & \phi(z), \\ \text{subject to} & z \in \mathcal{Z}. \end{array} \tag{4}$$

As discussed in [8, Chapter 9], ADMM can be used as a heuristic to solve non-convex constrained problems. ADMM has the form

$$\begin{aligned} w^{k+1} &:= \operatorname{argmin}_z(\phi(z) + (\rho/2)\|z - z^k + u^k\|_2^2) \\ z^{k+1} &:= \Pi(w^{k+1} + u^k) \\ u^{k+1} &:= u^k + w^{k+1} - z^{k+1}, \end{aligned} \tag{5}$$

where $\rho > 0$ is an algorithm parameter, k is the iteration counter, and Π denotes Euclidean projection onto $\mathcal{Z}$ (which need not be unique when $\mathcal{Z}$ is not convex).

The initial values u^0 and z^0 are additional algorithm parameters. We always set $u^0 = 0$ and draw z^0 randomly from a normal distribution $\mathcal{N}(0, \sigma^2 I)$, where $\sigma > 0$ is an algorithm parameter.

3.2 *Algorithm subroutines*

Convex proximal step. Carrying out the first step of the algorithm, that is, evaluating the proximal operator of ϕ, involves solving the convex optimization problem

$$\begin{array}{ll} \text{minimize} & f_0(x,z) + (\rho/2)\|z - z^k + u^k\|_2^2 \\ \text{subject to} & f_i(x,z) \leq 0,\ i = 1, \ldots, m, \\ & Ax + Bz = c, \end{array} \tag{6}$$

over the variables $x \in \mathbf{R}^n$ and $z \in \mathbf{R}^q$. This is the original problem (1), with the non-convex constraint $z \in \mathcal{Z}$ removed, and an additional convex quadratic term involving z added to the objective. We let (x^{k+1}, w^{k+1}) denote a solution of (6). If the problem (6) is infeasible, then so is the original problem (1); should this happen, we can terminate the algorithm with the certain conclusion that (1) is infeasible.

Projection. The (non-convex) projection step consists of finding a closest point in $\mathcal{Z}$ to $w^{k+1} + u^k$. If more than one point has the smallest distance, we can choose one of the minimizers arbitrarily. In cases where the projection onto $\mathcal{Z}$ is too costly, we replace projection with approximate projection.

Dual update. The iterate $u^k \in \mathbf{R}^q$ can be interpreted as a scaled dual variable, or as the running sum of the error values $w^k - z^k$.

3.3 *Discussion*

Convergence. When $\mathcal{Z}$ is convex (and a solution of (1) exists), this algorithm is guaranteed to converge to a solution, in the sense that $f_0(x^{k+1}, w^{k+1})$ converges to the optimal value of the problem (1), and $w^{k+1} - z^{k+1} \to 0$, that is, $w^{k+1} \to \mathcal{Z}$. See [8, §3] and the references therein for a more technical description and details. But in the general case, when $\mathcal{Z}$ is not convex, the algorithm is not guaranteed to converge, and even when it does, it need not be to a global, or even local, minimum. Some recent progress has been made on understanding convergence of ADMM in the non-convex case [56].

Parameters. Another difference with the convex case is that the convergence and the quality of solution depends on ρ, whereas for convex problems this algorithm is guaranteed to converge to the optimal value regardless of the choice of ρ. In other words, in the convex case the choice of parameter ρ only affects the speed of the convergence, while in the non-convex case the choice of ρ can have a critical role in the quality of approximate solution, as well as the speed of convergence.

The optimal parameter selection for ADMM is still an active research area in the convex case; even less is known about it in the non-convex case. In [31] the optimal parameter selection for convex quadratic problems is discussed. In a more general setting, Giselsson discusses the optimal parameter selection for ADMM for strongly convex functions in [32–34]. The dependence of global and local convergence properties of ADMM on parameter choice has been studied in [6,45].

Initialization. In the convex case the choice of initial point z^0 affects the number of iterations to find a solution, but not the quality of the solution. Unsurprisingly, the non-convex case differs in that the choice of z^0 has a major effect on the quality of the approximate solution. As with the choice of ρ, the initialization in the non-convex case is currently an active area of research; see, for example, [48,56,81]. A reasonable generic method is to draw initial points randomly from $\mathcal{N}(0, \sigma^2 I)$ (assuming reasonable scaling of the original problem).

3.4 *Solution improvement*

Now we describe two techniques to obtain better solutions after carrying out ADMM. The first technique relies on iterated neighbour search and the second one uses multiple restarts with random initial points in order to increase the chance of obtaining a better solution.

Iterated neighbour search. After each iteration, we can carry out iterated neighbour search (as described in Section 2.3) with $\mathcal{Z}^{\text{ngbr}}(z^{k+1}, D)$ to obtain $(\hat{x}^{k+1}, \hat{z}^{k+1})$. We will return the pair with the smallest merit function as the output of the algorithm. The distance D is a parameter that can be increased so that the neighbour search considers more points.

Multiple restarts. We choose the initial value z^0 from a normal distribution $\mathcal{N}(0, \sigma^2 I)$. We can run the algorithm multiple times from different initial points to increase the chance of finding a feasible point with a smaller objective value.

3.5 *Overall algorithm*

The following is a summary of the algorithm with solution improvement.

Algorithm 5 NC-ADMM heuristic

```
1: Initialize $u^0 = 0$, $(x_{\text{best}}, z_{\text{best}}) = \emptyset$, $\eta_{\text{best}} = \infty$.
2: for algorithm repeats $1, 2, \ldots, M$ do
3:     Initialize $z^0 \sim \mathcal{N}(0, \sigma^2 I)$, $k = 0$.
4:     do
5:         $(x^{k+1}, w^{k+1}) \leftarrow \operatorname{argmin}_z (\phi(z) + (\rho/2)\|z - z^k + u^k\|_2^2)$.
6:         $z^{k+1} \leftarrow \Pi(w^{k+1} + u^k)$.
7:         $u^{k+1} \leftarrow u^k + w^{k+1} - z^{k+1}$.
8:         Use Algorithm 4 on $(x^{k+1}, z^{k+1})$ to get the improved iterate $(\hat{x}, \hat{z})$.
9:         if $\eta(\hat{x}, \hat{z}) < \eta_{\text{best}}$ then
10:            $(x_{\text{best}}, z_{\text{best}}) \leftarrow (\hat{x}, \hat{z})$, $\eta_{\text{best}} = \eta(\hat{x}, \hat{z})$.
11:        end if
12:        $k \leftarrow k + 1$.
13:    while $k \leq N$ and $(\hat{x}, \hat{z})$ has not repeated $P$ times in a row.
14: end for
15: return $x_{\text{best}}, z_{\text{best}}$.
```

Convergence, stopping criteria, and optimality. As described in Section 3.3, ADMM need not converge for arbitrary non-convex $\mathcal{Z}$. The output of our heuristic is not the direct output of ADMM, however, but the output of ADMM after local search. In Algorithm 5, ADMM may be viewed as a procedure for generating sample points, which we run through Algorithm 4 to get different local optima. Our heuristic may therefore be useful even on problems where ADMM fails to converge. We terminate Algorithm 5 when local search returns the same point P times in a row, where P is a parameter. Given the lack of convergence guarantees for ADMM with non-convex $\mathcal{Z}$, the only formal notion of optimality provided by our heuristic is that the solution is optimal among all points considered by the local search method.

4. Projections onto non-convex sets

In this section we catalog various non-convex sets with their implied convex constraints, which will be included in the convex constraints of problem (1). We also provide a Euclidean projection (or approximate projection) Π for these sets. Also, when applicable, we introduce a non-trivial restriction and distance function.

4.1 *Subsets of* **R**

Booleans. For $\mathcal{Z} = \{0, 1\}$, a convex relaxation (in fact, the convex hull of $\mathcal{Z}$) is $[0, 1]$. Projection is simple rounding: $\Pi(z) = 0$ for $z \leq \frac{1}{2}$, and $\Pi(z) = 1$ for $z > \frac{1}{2}$. ($z = \frac{1}{2}$ can be mapped to either point.) Moreover, $\mathcal{Z}^{\text{dist}}(y, z) = |y - z|$ for $y, z \in \mathcal{Z}$.

Finite sets. If $\mathcal{Z}$ has M elements, the convex hull of $\mathcal{Z}$ is the interval from the smallest to the largest element. We can project onto $\mathcal{Z}$ with no more than $\log_2 M$ comparisons. For $y, z \in \mathcal{Z}$, the distance function is given by $\mathcal{Z}^{\text{dist}}(y, z) = |[y, z] \cap \mathcal{Z}| - 1$.

4.2 Subsets of $\mathbf{R}^n$

Boolean vectors with fixed cardinality. Let $\mathcal{Z} = \{z \in \{0,1\}^n \mid \mathrm{card}(z) = k\}$. Any $z \in \mathcal{Z}$ satisfies $0 \leq z \leq 1$ and $\mathbf{1}^T z = k$. We can project $z \in \mathbf{R}^n$ onto $\mathcal{Z}$ by setting the k entries of z with largest value to one and the remaining entries to zero. For $y,z \in \mathcal{Z}$, the distance $\mathcal{Z}^{\mathrm{dist}}(y,z)$ is defined as the minimum number of swaps of entries needed to transform y into z, or half the Hamming distance.

Vectors with bounded cardinality. Let $\mathcal{Z} = \{x \in [-M,M]^n \mid \mathrm{card}(x) \leq k\}$, where $M > 0$ and $k \in \mathbf{Z}_+$. (Vectors $z \in \mathcal{Z}$ are called k-sparse.) Any point $z \in \mathcal{Z}$ satisfies $-M \leq z \leq M$ and $-Mk \leq \mathbf{1}^T z \leq Mk$. The projection $\Pi(z)$ is found as follows

$$(\Pi(z))_i = \begin{cases} M & z_i > M,\ i \in \mathcal{I}, \\ -M & z_i < -M,\ i \in \mathcal{I}, \\ z_i & |z_i| \leq M,\ i \in \mathcal{I}, \\ 0 & i \notin \mathcal{I}, \end{cases}$$

where $\mathcal{I} \subseteq \{1,\ldots,n\}$ is a set of indices of k largest values of $|z_i|$. A restriction of $\mathcal{Z}$ at $z \in \mathcal{Z}$ is the set of all points in $[-M,M]^n$ that have the same sparsity pattern as z.

Quadratic sets. Let $\mathbf{S}^n_+$ and $\mathbf{S}^n_{++}$ denote the set of $n \times n$ symmetric positive semidefinite and symmetric positive definite matrices, respectively. Consider the set

$$\mathcal{Z} = \{z \in \mathbf{R}^n \mid \alpha \leq z^T A z + 2b^T z \leq \beta\},$$

where $A \in \mathbf{S}^n_{++}$, $b \in \mathbf{R}^n$, and $\beta \geq \alpha \geq -b^T A^{-1} b$. We assume $\alpha \geq -b^T A^{-1} b$ because $z^T A z + 2b^T z \geq -b^T A^{-1} b$ for all $z \in \mathbf{R}^n$. Any point $z \in \mathcal{Z}$ satisfies the convex inequality $z^T A z + 2b^T z \leq \beta$.

We can find the projection onto $\mathcal{Z}$ as follows. If $z^T A z + 2b^T z > \beta$, it suffices to solve

$$\begin{array}{ll} \text{minimize} & \|x - z\|_2^2, \\ \text{subject to} & x^T A x + 2b^T x \leq \beta, \end{array} \tag{7}$$

and if $z^T A z + 2b^T z < \alpha$, it suffices to solve

$$\begin{array}{ll} \text{minimize} & \|x - z\|_2^2 \\ \text{subject to} & x^T A x + 2b^T x \geq \alpha. \end{array} \tag{8}$$

(If $\alpha \leq z^T A z + 2b^T z \leq \beta$, clearly $\Pi(z) = z$.) The first problem is a convex quadratically constrained quadratic program and the second problem can be solved by solving a simple semidefinite program as described in [9, Appendix B]. Furthermore, there is a more efficient way to find the projection by finding the roots of a single-variable polynomial of degree $2p+1$, where p is the number of distinct eigenvalues of A [42,48]. Note that the projection can be easily found even if A is not positive definite; we assume $A \in \mathbf{S}^n_{++}$ only to make $\mathcal{Z}$ bounded and have a useful convex relaxation.

A restriction of $\mathcal{Z}$ at $z \in \mathcal{Z}$ is the set

$$\mathcal{Z}^{\mathrm{rstr}}(z) = \left\{ x \in \mathbf{R}^n \,\middle|\, \frac{x^T A z + b^T(x+z) + b^T A^{-1} b}{\sqrt{z^T A z + 2b^T z + b^T A^{-1} b}} \geq \sqrt{\alpha + b^T A^{-1} b},\ x^T A x + 2b^T x \leq \beta \right\}.$$

Recall that $z^T A z + 2b^T z + b^T A^{-1} b \geq 0$ for all $z \in \mathbf{R}^n$ and we assume $\alpha \geq -b^T A^{-1} b$, so $\mathcal{Z}^{\mathrm{rstr}}(z)$ is always well defined.

Annulus and sphere. Consider the set

$$\mathcal{Z} = \{z \in \mathbf{R}^n \mid r \le \|z\|_2 \le R\},$$

where $R \ge r$.

Any point $z \in \mathcal{Z}$ satisfies $\|z\|_2 \le R$. We can project $z \in \mathbf{R}^n \setminus \{0\}$ onto $\mathcal{Z}$ by the following scaling

$$\Pi(z) = \begin{cases} rz/\|z\|_2 & \text{if } \|z\|_2 < r, \\ z & \text{if } z \in \mathcal{Z}, \\ Rz/\|z\|_2 & \text{if } \|z\|_2 > R. \end{cases}$$

If $z = 0$, any point with Euclidean norm r is a valid projection.

A restriction of $\mathcal{Z}$ at $z \in \mathcal{Z}$ is the set

$$\mathcal{Z}^{\text{rstr}}(z) = \{x \in \mathbf{R}^n \mid x^{\mathrm{T}} z \ge r\|z\|_2,\ \|x\|_2 \le R\}.$$

Notice that if $r = R$, then $\mathcal{Z}$ is a sphere and the restriction will be a singleton.

4.3 *Subsets of* $\mathbf{R}^{m\times n}$

Remember that the projection of a point $X \in \mathbf{R}^{m\times n}$ on a set $\mathcal{Z} \subset \mathbf{R}^{m\times n}$ is a point $Z \in \mathcal{Z}$ such that the Frobenius norm $\|X - Z\|_{\mathrm{F}}$ is minimized. As always, if there is more than one point Z that minimizes $\|X - Z\|_{\mathrm{F}}$, we accept any of them.

Matrices with bounded singular values and orthogonal matrices. Consider the set of $m \times n$ matrices whose singular values lie between 1 and α

$$\mathcal{Z} = \{Z \in \mathbf{R}^{m\times n} \mid I \preceq Z^{\mathrm{T}} Z \preceq \alpha^2 I\},$$

where $\alpha \ge 1$, and $A \preceq B$ means $B - A \in \mathbf{S}^n_+$. Any point $Z \in \mathcal{Z}$ satisfies $\|Z\|_2 \le \alpha$.

If $Z = U\Sigma V^{\mathrm{T}}$ is the singular value decomposition of Z with singular values $(\sigma_z)_{\min\{m,n\}} \le \cdots \le (\sigma_z)_1$ and $X \in \mathcal{Z}$ with singular values $(\sigma_x)_{\min\{m,n\}} \le \cdots \le (\sigma_x)_1$, according to the von Neumann trace inequality [87] we will have

$$\mathrm{Tr}(Z^{\mathrm{T}} X) \le \sum_{i=1}^{\min\{m,n\}} (\sigma_z)_i (\sigma_x)_i.$$

Hence

$$\|Z - X\|_F^2 \ge \sum_{i=1}^{\min\{m,n\}} ((\sigma_z)_i - (\sigma_x)_i)^2,$$

with equality when $X = U\mathrm{diag}(\sigma_x)V^{\mathrm{T}}$. This inequality implies that $\Pi(Z) = U\tilde{\Sigma}V^{\mathrm{T}}$, where $\tilde{\Sigma}$ is a diagonal matrix and $\tilde{\Sigma}_{ii}$ is the projection of Σ_{ii} on the interval $[1, \alpha]$. When $Z = 0$, the projection $\Pi(Z)$ is any matrix.

Given $Z = U\Sigma V^{\mathrm{T}} \in \mathcal{Z}$, we have the following restriction [7]

$$\mathcal{Z}^{\text{rstr}}(Z) = \{X \in \mathbf{R}^{m\times n} \mid \|X\|_2 \le \alpha,\ V^{\mathrm{T}} X^{\mathrm{T}} U + U^{\mathrm{T}} X V \succeq 2I\}.$$

(Notice that $X \in \mathcal{Z}^{\text{rstr}}(Z)$ satisfies $X^{\mathrm{T}} X \succeq I + (X - UV^{\mathrm{T}})^{\mathrm{T}}(X - UV^{\mathrm{T}}) \succeq I$.)

There are several noteworthy special cases. When $\alpha = 1$ and $m = n$ we have the set of orthogonal matrices. In this case, the restriction will be a singleton. When $n = 1$, the set $\mathcal{Z}$ is equivalent to the annulus $\{z \in \mathbf{R}^m \mid 1 \le \|z\|_2 \le \alpha\}$.

Matrices with bounded rank. Let $\mathcal{Z} = \{Z \in \mathbf{R}^{m\times n} \mid \mathrm{Rank}(Z) \le k,\ \|Z\|_2 \le M\}$. Any point $Z \in \mathcal{Z}$ satisfies $\|Z\|_2 \le M$ and $\|Z\|_* \le Mk$, where $\|\cdot\|_*$ denotes the trace norm. If $Z = U\Sigma V^{\mathrm{T}}$ is the singular value decomposition of Z, we will have $\Pi(Z) = U\tilde{\Sigma}V^{\mathrm{T}}$, where $\tilde{\Sigma}$ is a diagonal matrix with $\tilde{\Sigma}_{ii} = \min\{\Sigma_{ii}, M\}$ for $i = 1,\dots k$, and $\tilde{\Sigma}_{ii} = 0$ otherwise.

Given a point $Z \in \mathcal{Z}$, we can write the singular value decomposition of Z as $Z = U\Sigma V^{\mathrm{T}}$ with $U \in \mathbf{R}^{m\times k}$, $\Sigma \in \mathbf{R}^{r\times r}$ and $V \in \mathbf{R}^{n\times k}$. A restriction of $\mathcal{Z}$ at Z is

$$\mathcal{Z}^{\mathrm{rstr}}(Z) = \{U\tilde{\Sigma}V^{\mathrm{T}} \mid \tilde{\Sigma} \in \mathbf{R}^{r\times r}\}.$$

Assignment and permutation matrices. *Assignment matrices* are Boolean matrices with exactly one non-zero element in each column and at most one non-zero element in each row. (They represent an assignment of the columns to the rows.) In other words, the set of assignment matrices on $\{0,1\}^{m\times n}$, where $m \ge n$, satisfy

$$\sum_{j=1}^{n} Z_{ij} \le 1, \quad i = 1,\dots,m,$$

$$\sum_{i=1}^{m} Z_{ij} = 1, \quad j = 1,\dots,n.$$

These two sets of inequalities, along with $0 \le Z_{ij} \le 1$ are the implied convex inequalities. When $m = n$, this set becomes the set of permutation matrices, which we denote by $\mathcal{P}_n$.

Projecting $Z \in \mathbf{R}^{m\times n}$ (with $m \ge n$) onto the set of assignment matrices involves choosing an entry from each column of Z such that no two chosen entries are from the same row and the sum of chosen entries is maximized. Assuming that the entries of Z are the weights of edges in a bipartite graph, the projection onto the set of assignment matrices will be equivalent to finding a maximum-weight matching in a bipartite graph. The Hungarian method [52] is a well-known polynomial time algorithm to find the maximum weight matching, and hence also the projection onto assignment matrices.

For $Y, Z \in \mathcal{Z}$, the distance $\mathcal{Z}^{\mathrm{dist}}(Y,Z)$ is defined as the minimum number of swaps of adjacent rows and columns necessary to transform Y into Z. We define distance in terms of swaps of adjacent rows and columns rather than arbitrary rows and columns to reduce the number of neighbours. For example, the restriction that swaps must be of adjacent rows and columns reduces $|\mathcal{Z}^{\mathrm{ngbr}}(Z,1)|$ from $O(mn)$ to $O(m+n)$ for $Z \in \mathcal{Z}$.

Hamiltonian cycles. A *Hamiltonian cycle* is a cycle in a graph that visits every node exactly once. Every Hamiltonian cycle in a complete graph can be represented by its adjacency matrix, for example

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

represents a Hamiltonian cycle that visits nodes $(3,2,4,1)$ sequentially. Let $\mathcal{H}_n$ be the set of $n \times n$ matrices that represent a Hamiltonian cycle.

Every point $Z \in \mathcal{H}_n$ satisfies $0 \le Z_{ij} \le 1$ for $i,j = 1,\dots,n$, and $Z = Z^{\mathrm{T}}$, $(1/2)Z\mathbf{1} = \mathbf{1}$, and

$$2\mathbf{I} - Z + 4\frac{\mathbf{1}\mathbf{1}^{\mathrm{T}}}{n} \succeq 2\left(1 - \cos\frac{2\pi}{n}\right)\mathbf{I},$$

where $\mathbf{I}$ denotes the identity matrix. In order to see why the last inequality holds, it is enough to note that $2\mathbf{I} - Z$ is the Laplacian of the cycle represented by Z [2,59]. It can be shown that the smallest eigenvalue of $2\mathbf{I} - Z$ is zero (which corresponds to the eigenvector $\mathbf{1}$), and the second smallest eigenvalue of $2\mathbf{I} - Z$ is $2(1 - \cos\frac{2\pi}{n})$. Hence all eigenvalues of $2\mathbf{I} - Z + 4(\mathbf{1}\mathbf{1}^{\mathrm{T}}/n)$ must be no smaller than $2(1 - \cos(2\pi/n))$.

We are not aware of a polynomial time algorithm to find the projection of a given real $n \times n$ matrix onto $\mathcal{H}_n$. We can find an *approximate projection* of Z by the following greedy algorithm: construct a graph with n vertices where the edge between i and j is weighted by z_{ij}. Start with the edge with largest weight and at each step, among all the edges that do not create a cycle, choose the edge with the largest weight (except for the last step where a cycle is created).

For $Y, Z \in \mathcal{H}_n$, the distance $\mathcal{Z}^{\mathrm{dist}}(Y, Z)$ is defined as the minimum number of adjacent nodes that must be swapped to transform Y into Z. Swapping adjacent nodes i and j means replacing Y with $P_{(i,j)}YP_{(i,j)}^{\mathrm{T}}$ where $Y_{ij} = 1$ and $P_{(i,j)}$ is a permutation matrix that swaps nodes i and j and leaves other nodes unchanged. As with assignment matrices, we define distance in terms of swaps of adjacent nodes rather than arbitrary nodes to reduce the number of neighbours.

4.4 *Combinations of sets*

Cartesian product. Let $\mathcal{Z} = \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_k \subset \mathbf{R}^n$, where $\mathcal{Z}_1, \ldots, \mathcal{Z}_k$ are closed sets with known projections (or approximate projections). A convex relaxation of $\mathcal{Z}$ is the Cartesian product $\mathcal{Z}_1^{\mathrm{rlx}} \times \cdots \times \mathcal{Z}_k^{\mathrm{rlx}}$, where $\mathcal{Z}_i^{\mathrm{rlx}}$ is the set described by the convex relaxation of $\mathcal{Z}_i$. The projection of $z \in \mathbf{R}^n$ onto $\mathcal{Z}$ is $(\Pi_1(z_1), \ldots, \Pi_k(z_k))$, where Π_i denotes the projection onto $\mathcal{Z}_i$ for $i = 1, \ldots, k$.

A restriction of $\mathcal{Z}$ at a point $z = (z_1, z_2, \ldots, z_k) \in \mathcal{Z}$ is the Cartesian product $\mathcal{Z}^{\mathrm{rstr}}(z) = \mathcal{Z}_1^{\mathrm{rstr}}(z_1) \times \cdots \times \mathcal{Z}_k^{\mathrm{rstr}}(z_k)$. For $y = (y_1, y_2, \ldots, y_k) \in \mathcal{Z}$ and $z = (z_1, z_2, \ldots, z_k) \in \mathcal{Z}$, the distance function is given by $\mathcal{Z}^{\mathrm{dist}}(y, z) = \sum_{i=1}^{k} \mathcal{Z}_i^{\mathrm{dist}}(y_i, z_i)$.

5. Implementation

We have implemented the NCVX Python package for modelling problems of the form (1) and applying the NC-ADMM heuristic, along with the relax–round–polish and relax methods. The NCVX package is an extension of CVXPY [22]. The problem objective and convex constraints are expressed using standard CVXPY semantics. Non-convex constraints are expressed implicitly by creating a variable constrained to lie in one of the sets described in Section 4. For example, the code snippet

```
x = Boolean()
```

creates a variable $x \in \mathbf{R}$ with the implicit non-convex constraint $x \in \{0, 1\}$. The convex relaxation, in this case $x \in [0, 1]$, is also implicit in the variable definition. The source code for NCVX is available at https://github.com/cvxgrp/ncvx.

5.1 *Variable constructors*

The NCVX package provides the following functions for creating variables with implicit non-convex constraints, along with many others not listed:

- `Boolean(n)` creates a variable $x \in \mathbf{R}^n$ with the implicit constraint $x \in \{0, 1\}^n$.
- `Integer(n, M)` creates a variable $x \in \mathbf{R}^n$ with the implicit constraints $x \in \mathbf{Z}^n$ and $\|x\|_\infty \leq \lfloor M \rfloor$.

- `Card(n, k, M)` creates a variable $x \in \mathbf{R}^n$ with the implicit constraints that at most k entries are non-zero and $\|x\|_\infty \leq M$.
- `Choose(n, k)` creates a variable $x \in \mathbf{R}^n$ with the implicit constraints that $x \in \{0,1\}^n$ and has exactly k non-zero entries.
- `Rank(m, n, k, M)` creates a variable $X \in \mathbf{R}^{m \times n}$ with the implicit constraints $\mathrm{Rank}(X) \leq k$ and $\|X\|_2 \leq M$.
- `Assign(m, n)` creates a variable $X \in \mathbf{R}^{m \times n}$ with the implicit constraint that X is an assignment matrix.
- `Permute(n)` creates a variable $X \in \mathbf{R}^{n \times n}$ with the implicit constraint that X is a permutation matrix.
- `Cycle(n)` creates a variable $X \in \mathbf{R}^{n \times n}$ with the implicit constraint that X is the adjacency matrix of a Hamiltonian cycle.
- `Annulus(n,r,R)` creates a variable $x \in \mathbf{R}^n$ with the implicit constraint $r \leq \|x\|_2 \leq R$.
- `Sphere(n, r)` creates a variable $x \in \mathbf{R}^n$ with the implicit constraint $\|x\|_2 = r$.

5.2 *Variable methods*

Additionally, each variable created by the functions in Section 5.1 supports the following methods:

- `variable.relax()` returns a list of convex constraints that represent a convex relaxation of the non-convex set $\mathcal{Z}$, to which the variable belongs.
- `variable.project(z)` returns the Euclidean (or approximate) projection of z onto the non-convex set $\mathcal{Z}$, to which the variable belongs.
- `variable.restrict(z)` returns a list of convex constraints describing the convex restriction $\mathcal{Z}^{\mathrm{rstr}}(z)$ at z of the non-convex set $\mathcal{Z}$, to which the variable belongs.
- `variable.neighbours(z, D)` returns a list of neighbours $\mathcal{Z}^{\mathrm{ngbr}}(z, D)$ of z contained in the non-convex set $\mathcal{Z}$, to which the variable belongs.

Users can add support for additional non-convex sets by providing functions that implement these four methods.

5.3 *Constructing and solving problems*

To construct a problem of the form (1), the user creates variables $z_1, \ldots, z_k$ with the implicit constraints $z_1 \in \mathcal{Z}_1, \ldots, z_k \in \mathcal{Z}_k$, where $\mathcal{Z}_1, \ldots, \mathcal{Z}_k$ are non-convex sets, using the functions described in Section 5.1. The variable z in problem (1) corresponds to the vector $(z_1, \ldots, z_k)$. The components of the variable x, the objective, and the constraints are constructed using standard CVXPY syntax.

Once the user has constructed a problem object, they can apply the following solve methods:

- `problem.solve(method="relax")` solves the convex relaxation of the problem.
- `problem.solve(method="relax-round-polish")` applies the relax–round–polish heuristic. Additional arguments can be used to specify the parameters N, D, σ, and λ. By default the parameter values are $N = 5$, $D = 1$, $\sigma = 1$, and $\lambda = 10^4$. When $N > 1$, the first sample $w_1 \in \mathbf{R}^q$ is always 0. Subsequent samples are drawn i.i.d. from $N(0, \sigma^2 I)$. Neighbour search looks at all neighbours within distance D.
- `problem.solve(method="nc-admm")` applies the NC-ADMM heuristic. Additional arguments can be used to specify the number of starting points, the number of iterations the algorithm is run from each starting point, and the values of the parameters ρ, D, σ, and λ. By

default the algorithm is run from 5 starting points for 50 iterations, the value of ρ is drawn uniformly from $[0,1]$, and the other parameter values are $D = 1$, $\sigma = 1$, and $\lambda = 10^4$. The first starting point is always $z^0 = 0$ and subsequent starting points are drawn i.i.d. from $\mathcal{N}(0, \sigma^2 I)$. Neighbour search looks at all neighbours within distance D.

The relax–round–polish and NC-ADMM methods record the best point found $(x_{\text{best}}, z_{\text{best}})$ according to the merit function. The methods return the objective value $f_0(x_{\text{best}}, z_{\text{best}})$ and the residual $r(x_{\text{best}}, z_{\text{best}})$, and set the `value` field of each variable to the appropriate segment of x_{best} and z_{best}.

For example, consider the *regressor selection* problem, which we will discuss in Section 6.1. This problem can be formulated as

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|_2^2 \\ \text{subject to} & \operatorname{card}(x) \le k, \quad \|x\|_\infty \le M, \end{array} \tag{9}$$

with decision variable $x \in \mathbf{R}^n$ and problem data $A \in \mathbf{R}^{m\times n}$, $b \in \mathbf{R}^m$, $M > 0$, and $k \in \mathbf{Z}_+$. The following code attempts to approximately solve this problem using our heuristic.

```
x = Card(n,k,M)
prob = Problem(Minimize(sum_squares(A*x-b)))
objective, residual = prob.solve(method="nc-admm")
```

The first line constructs a variable $x \in \mathbf{R}^n$ with the implicit constraints that at most k entries are non-zero, $\|x\|_\infty \le M$, and $\|x\|_1 \le kM$. The second line creates a minimization problem with objective $\|Ax - b\|_2^2$ and no constraints. The last line applies the NC-ADMM heuristic to the problem and returns the objective value and residual of the best point found.

5.4 *Limitations*

Our implementation is designed to be simple and to generalize to as many problems as possible. As a result, the implementation has several limitations in terms of computational efficiency and exploiting problem specific structure. For example, no work is cached across solves of convex subproblems. Caching factorizations or warm starting would improve performance when the convex solver supports these features. The implementation runs NC-ADMM from different initial values in parallel, but a more sophisticated implementation would use finer grained parallelism. Neighbour search and polishing can be made more efficient than the general purpose approach in our implementation by exploiting problem specific structure. For example, if the variable z is a Boolean vector, that is, $\mathcal{Z} \in \{0,1\}^q$, then any neighbour $\tilde{z} \in \mathcal{Z}^{\text{rstr}}(z, 1)$ differs from z in only two entries. The change in the merit function $\eta(x, \tilde{z}) - \eta(x, z)$ can be computed efficiently given $\eta(x, z)$, accelerating neighbour search. Polishing can also be accelerated by taking advantage of the structure of the convex restriction. For instance, the restriction of the cardinality constraint $\mathcal{Z} = \{z \in \mathbf{R}^q \mid \operatorname{card}(z) \le k,\ \|z\|_\infty \le M\}$ fixes the sparsity pattern, which reduces the number of free entries of z from q to k. Our implementation imposes the restriction by adding convex constraints to the problem, but a more efficient implementation would replace z with a k-dimensional variable.

For several of the examples in Section 6, we implemented optimized versions of the NC-ADMM algorithm that remedy the limitations of our general purpose implementation. Even our problem specific implementations could be improved further by better exploiting parallelism and applying low-level code optimization, but the implementations are fast enough to compete with optimized general purpose mixed integer solvers like Gurobi [37].

6. Examples

In this section we apply the NC-ADMM heuristic to a wide variety of hard problems, that is, that generally cannot be solved in polynomial time. Extensive research has been done on specialized algorithms for each of the problems discussed in this section. Our intention is not to seek better performance than these specialized algorithms, but rather to show that our general purpose heuristic can yield decent results with minimal tuning. The advantage of our heuristic is that it can be applied to problems that no one has studied before, not that it outperforms the state-of-the-art on well-studied problems.

Unless otherwise specified, the algorithm parameters are the defaults described in Section 5. In particular, we use random initialization for all examples. For most problems a well chosen problem specific initialization will improve the results of our method; see, for example [48,56,81]. We use random initialization, however, because it better demonstrates that our heuristic can be effective with minimal tuning. Whenever possible, we compare our heuristic to Gurobi [37], a commercial global optimization solver. All runtimes reported are on a laptop with a four-core 2.3 GHz Intel Core i7 processor.

6.1 *Regressor selection*

We consider the problem of approximating a vector b with a linear combination of at most k columns of A with bounded coefficients. This problem can be formulated as

$$\begin{array}{ll} \text{minimize} & \|Ax-b\|_2^2 \\ \text{subject to} & \operatorname{card}(x) \le k, \quad \|x\|_\infty \le M, \end{array} \tag{10}$$

with decision variable $x \in \mathbf{R}^n$ and problem data $A \in \mathbf{R}^{m\times n}$, $b \in \mathbf{R}^m$, $k \in \mathbf{Z}_+$, and $M > 0$. Lasso (least absolute shrinkage and selection operator) is a well-known heuristic for solving this problem by adding ℓ_1 regularization and minimizing $\|Ax-b\|_2^2 + \lambda\|x\|_1$. The value of λ is chosen as the smallest value for which $\operatorname{card}(x) \le k$ (see [29, §3.4] and [9, §6.3]). The nonconvex set from Section 4 in problem (10) is the set of vectors with bounded cardinality $\mathcal{Z} = \{x \in [-M,M]^n \mid \operatorname{card}(x) \le k\}$.

Problem instances. We first consider a family of random problem instances. We generated the matrix $A \in \mathbf{R}^{m\times 2m}$ with i.i.d. $\mathcal{N}(0,1)$ entries, and chose $b = A\hat{x} + v$, where $\hat{x}$ was drawn uniformly at random from the set of vectors satisfying $\operatorname{card}(\hat{x}) \le \lfloor m/5 \rfloor$ and $\|x\|_\infty \le 1$, and $v \in \mathbf{R}^m$ was a noise vector drawn from $\mathcal{N}(0,\sigma^2 I)$. We set $\sigma^2 = \|A\hat{x}\|^2/(400m)$ so that the signal-to-noise ratio was near 20. For each value of m, we generated 40 instances of the problem as described above. We solved the instances for $k = \lfloor m/5 \rfloor$.

In order to examine this method on a real data set, we also used data from the University of California, Irvine (UCI) Machine Learning repository [28] to study the murder rate (per 100K people) of $m = 2215$ communities in the United States. Similar to [83], we had $n = 101$ attributes measured in each community, and our goal was to predict the murder rate as a linear function of only k attributes. To find a good prediction model one would use cross validation analysis in order to choose k; but we limited ourselves to the problem of finding $2 \le k \le 20$ regressors that minimize $\|Ax-b\|_2^2$.

Results. Figure 1 compares the average sum of squares error for the x^* values found by the Lasso heuristic, relax–round–polish, and NC-ADMM for the randomly generated instances. For Lasso, we solved the problem for 100 values of λ and then solved the polishing problem after

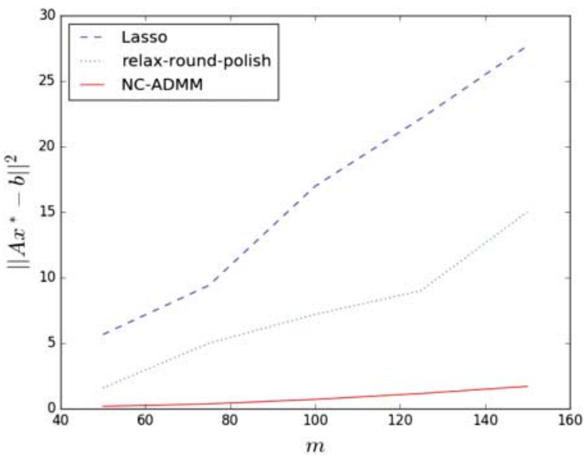

Figure 1. The average error of solutions found by Lasso, relax–round–polish, and NC-ADMM for 40 random instances of the regressor selection problem.

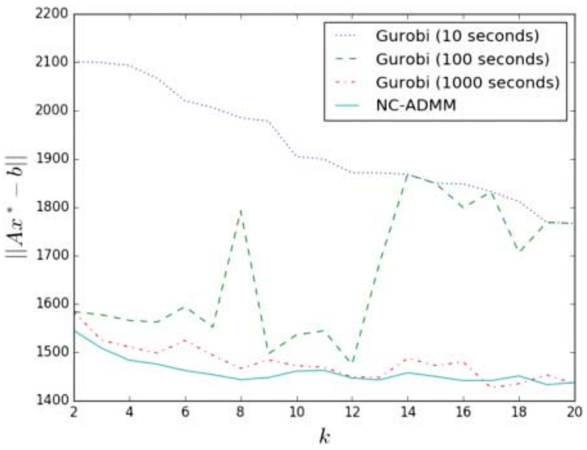

Figure 2. The best value found by NC-ADMM (usually done in 35 milliseconds) and Gurobi after 10, 100, and 1000 s.

fixing the sparsity pattern suggested by Lasso. For all m, the objective values found by the NC-ADMM heuristic were on average better than those found by the Lasso and relax–round–polish heuristics.

For our second problem (murder rate), we used our NC-ADMM heuristic and Gurobi to solve the problem. Our tailored implementation of NC-ADMM never took more than 40 milliseconds to run. The implementation is extremely efficient because the dominant computation is a single factorization of the matrix $A^TA + \rho I$. We use only one restart and hence only one value of ρ. Figure 2 shows the value found by NC-ADMM as well as the best value found by Gurobi after 10, 100, and 1000 s. For all k, the objective value found by NC-ADMM after only 40 milliseconds was better than those found by Gurobi after 10 or 100 s and comparable to those found after 1000 s. (Of course, Gurobi will eventually find the global optimal point, and therefore match or beat the point found by NC-ADMM.)

6.2 *3-satisfiability*

Given Boolean variables $x_1, \ldots, x_n$, a *literal* is either a variable or the negation of a variable, for example x_1 and $\neg x_2$. A *clause* is disjunction of literals (or a single literal), for example $(\neg x_1 \vee$

$x_2 \vee \neg x_3$). Finally a formula is in conjunctive normal form (CNF) if it is a conjunction of clauses (or a single clause), for example $(\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2)$. Determining the satisfiability of a formula in CNF where each clause is limited to at most three literals is called *3-satisfiability* or simply the *3-SAT* problem. It is known that 3-SAT is NP-complete, hence we do not expect to be able to solve 3-SAT in general using our heuristic. A 3-SAT problem can be formulated as the following

$$\begin{array}{ll} \text{minimize} & 0, \\ \text{subject to} & Az \leq b, \\ & z \in \{0,1\}^n, \end{array} \tag{11}$$

where entries of $A \in \mathbf{R}^{m \times n}$ are given by

$$a_{ij} = \begin{cases} -1 & \text{if clause } i \text{contains } x_j, \\ 1 & \text{if clause } i \text{contains} \neg x_j, \\ 0 & \text{otherwise,} \end{cases}$$

and the entries of b are given by

$$b_i = (\text{number of negated literals in clause } i) - 1.$$

The non-convex set from Section 4 in problem (11) is the set of Boolean vectors $\mathcal{Z} = \{0,1\}^n$.

Problem instances. We generated 3-SAT problems with varying numbers of clauses and variables randomly as in [58,60]. As discussed in [16], there is a threshold around 4.25 clauses per variable when problems transition from being feasible to being infeasible. Problems near this threshold are generally found to be hard satisfiability problems. The SATLIB uniform random-3-SAT benchmark is constructed by the same method [47]. We generated 10 instances for each choice of number of clauses m and variables n, verifying that each instance is feasible using Gurobi [37].

Results. We ran the NC-ADMM heuristic on each instance, with 10 restarts and 100 iterations, and $\rho = 10$. Figure 3 shows the fraction of instances solved correctly with NC-ADMM for each choice of number of clauses m and variables n. We see that using this heuristic, satisfying assignments can be found consistently for up to 3.2 constraints per variable, at which point success starts to decrease. Problems in the gray region in Figure 3 were not tested since they are infeasible with high probability. We also tried the relax–round–polish heuristic, but it often failed to solve problems with more than 50 clauses.

For all instances, the runtime of the NC-ADMM heuristic with the parameters we chose was greater than the time it took Gurobi to find a solution. A specialized SAT solver would of course be even faster. We include the example nonetheless because it shows that the NC-ADMM heuristic can be effective for feasibility problems, even though the algorithm is not guaranteed to find a feasible point.

6.3 *Circle packing*

In the *circle packing problem* we are interested in finding the smallest square in which we can place n non-overlapping circles with radii $r_1, \ldots, r_n$ [36]. This problem has been studied extensively [11,14,79] and a database of densest known packings (with all r_i equal) for different

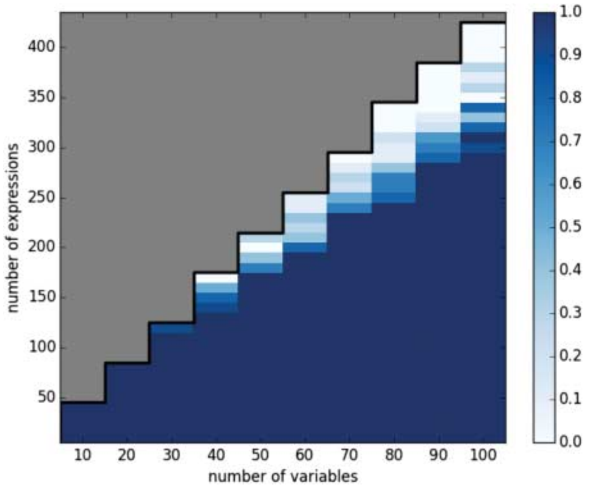

Figure 3. The fraction of the 10 3-SAT instances generated for each choice of number of clauses m and variables n for which NC-ADMM found a satisfying assignment. No instances were generated for (n, m) in the gray region.

numbers of circles can be found in [77]. Variants of the problem arise in industrial packing and computer aided design [41]. The problem can be formulated as

$$\begin{array}{ll} \text{minimize} & l \\ \text{subject to} & r_i\mathbf{1} \le x_i \le (l - r_i)\mathbf{1}, \quad i = 1, \dots, n \\ & x_i - x_j = z_{ij}, \quad i = 1, \dots, n-1,\ j = i+1, \dots, n \\ & 2\sum_{k=1}^{n} r_i \ge \|z_{ij}\|_2 \ge r_i + r_j, \quad i = 1, \dots, n-1,\ j = i+1, \dots, n, \end{array} \tag{12}$$

where $x_1, \dots, x_n \in \mathbf{R}^2$ are variables representing the circle centres and $z_{12}, z_{13}, \dots, z_{n-1,n} \in \mathbf{R}^2$ are additional variables representing the offset between pairs (x_i, x_j). The non-convex set from Section 4 in problem (12) are the annuli

$$\mathcal{Z}_{ij} = \left\{ z_{ij} \in \mathbf{R}^2 \mid r_i + r_j \le \|z_{ij}\|_2 \le 2\sum_{k=1}^{n} r_i \right\},$$

for $i = 1, \dots, n-1$ and $j = i+1, \dots, n$.

Problem instances. We generated problems with different numbers of circles n, but with equal radii $r_1, \dots, r_n$. Problem instances of this form are quite difficult to solve globally. The densest possible packing is unknown for most $n > 36$ [77].

Results. We ran the relax–round–polish heuristic for problems with $n = 1, \dots, 100$. The heuristic is essentially equivalent to well-known methods like the convex-concave procedure and the majorization–minimization algorithm [58]. We observed that NC-ADMM is no more effective than relax–round–polish. Figure 4 shows the relative radius r_1/l of the packing found by our heuristic in comparison to the best packing known. Figure 5 shows the packing found by our heuristic for $n = 41$. The obtained packing covers 78.68% of the area of the bounding square, which is close to the densest known packing, which covers 79.27% of the area.

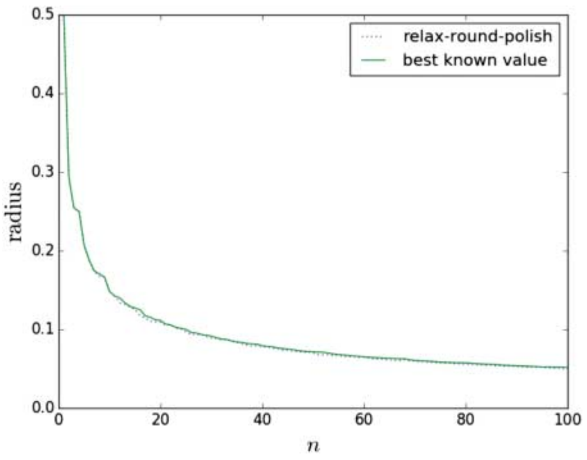

Figure 4. The relative radius r_1/l for the densest known packing and the packing found with the relax–round–polish heuristic for $n = 1, \dots, 100$.

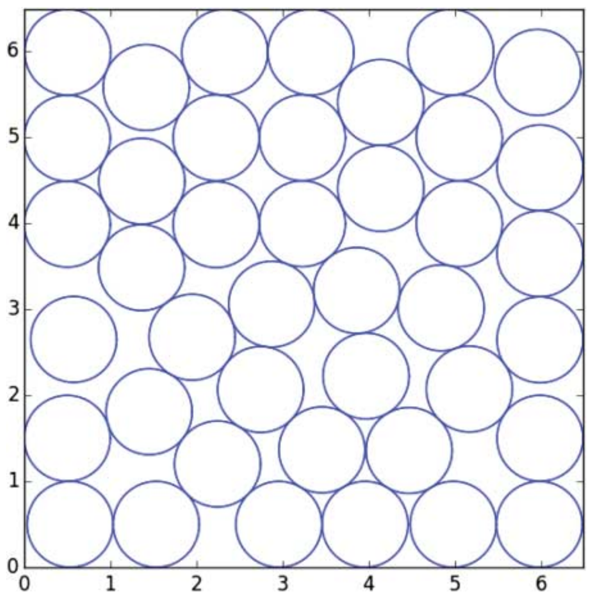

Figure 5. The packing for $n = 41$ circles with equal radii found with the relax–round–polish heuristic.

6.4 *Travelling salesman problem*

In the TSP, we wish to find the minimum weight Hamiltonian cycle in a weighted graph. A Hamiltonian cycle is a path that starts and ends on the same vertex and visits each other vertex in the graph exactly once. Let G be a graph with n vertices and $D \in \mathbf{S}^n$ be the (weighted) adjacency matrix, that is, the real number d_{ij} denotes the distance between i and j. We can formulate the TSP problem for G as follows

$$\begin{array}{ll} \text{minimize} & (1/2)\operatorname{Tr}(D^{\mathrm{T}}Z), \\ \text{subject to} & Z \in \mathcal{H}_n, \end{array} \tag{13}$$

where Z is the decision variable [18,43,51,53]. The non-convex set from Section 4 in problem (13) is the set of Hamiltonian cycles $\mathcal{Z} = \mathcal{H}_n$.

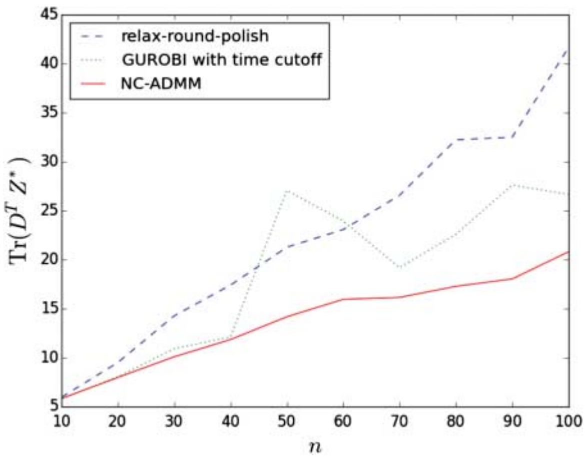

Figure 6. The average cost of the TSP solutions found by relax–round–polish, NC-ADMM, and Gurobi with a time cutoff equal to the runtime of NC-ADMM.

Problem instances. We generated problems with different numbers of vertices n by sampling n points from the uniform distribution on $[-1, 1]^2$. We set d_{ij} to be the Euclidean distance between points i and j. For each value of n, we generated 10 instances of the problem according to the above procedure.

Results. Figure 6 shows the average cost of the solutions found by relax–round–polish, NC-ADMM, and Gurobi with a time cutoff. We implemented an optimized version of NC-ADMM for the TSP problem. We ran NC-ADMM with 4 restarts and 25 iterations. We ran Gurobi on the standard MILP formulation of the TSP [65, §13] and gave it a time cutoff equal to the runtime of our NC-ADMM implementation. We ignored instances where Gurobi failed to find a feasible point within the runtime.

As n increases, the average cost of the solutions found by NC-ADMM goes below that of Gurobi with a time cutoff. Of course a specialized TSP solver like Concorde [3] could solve all the problem instances to global optimality within the runtime of NC-ADMM. We emphasize again, however, that our goal is not to outperform specialized solvers on every problem class, but simply for NCVX to compare favourably with other general purpose non-convex solvers.

6.5 *Factor analysis model*

The factor analysis problem decomposes a matrix as a sum of a low-rank and a diagonal matrix and has been studied extensively (e.g. [63,70]). It is also known as the *Frisch* scheme in the system identification literature [19,50]. The problem is the following

$$\begin{array}{ll} \text{minimize} & \|\Sigma - \Sigma^{\mathrm{lr}} - D\|_F^2 \\ \text{subject to} & D = \mathrm{diag}(d), \quad d \geq 0 \\ & \Sigma^{\mathrm{lr}} \succeq 0 \\ & \mathrm{Rank}(\Sigma^{\mathrm{lr}}) \leq k, \end{array} \tag{14}$$

where $\Sigma^{\mathrm{lr}} \in \mathbf{S}_+^n$ and diagonal matrix $D \in \mathbf{R}^{n\times n}$ with non-negative diagonal entries are the decision variables, and $\Sigma \in \mathbf{S}_+^n$ and $k \in \mathbf{Z}_+$ are problem data. One well-known heuristic for solving

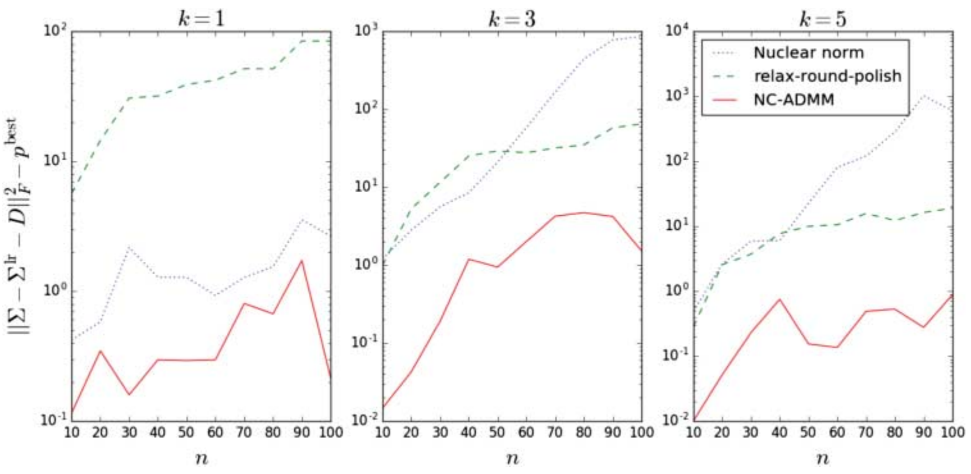

Figure 7. The average difference between the objective value found by the nuclear norm, relax–round–polish, and NC-ADMM heuristics and the best objective value found by any of the heuristics for instances of the factor analysis problem constructed from daily stock returns.

this problem is adding $\|\cdot\|_*$, or nuclear norm, regularization and minimizing $\|\Sigma - \Sigma^{\rm lr} - D\|_F^2 + \lambda\|\Sigma^{\rm lr}\|_*$ [70,85]. The value of λ is chosen as the smallest value possible such that $\mathrm{Rank}(\Sigma^{\rm lr}) \leq k$. Since $\Sigma^{\rm lr}$ is positive semidefinite, $\|\Sigma^{\rm lr}\|_* = \mathrm{Tr}(\Sigma^{\rm lr})$. The non-convex set from Section 4 for problem (14) is the set of matrices with bounded rank $\mathcal{Z} = \{\Sigma^{\rm lr} \in \mathbf{S}^n_+ \mid \mathrm{Rank}(\Sigma^{\rm lr}) \leq k\}$. Unlike in Section 4, we constrain $\Sigma^{\rm lr}$ to be positive semidefinite but impose no bound on the norm $\|\Sigma^{\rm lr}\|_2$.

Problem instances. We constructed instances of the factor analysis problem using daily returns from stocks in the July 2016 S&P 500 over 2014 and 2015. There is a long history in finance of decomposing the covariance of stock returns into low-rank and diagonal components [67,75]. We varied the number of stocks used n, the rank k, and the month of returns history considered. For each choice of n, k, and month, we generated an instance of problem (14) by setting Σ to be the covariance matrix of the daily per cent returns over that month for the first n S&P 500 stocks, ordered alphabetically by NYSE ticker.

Results. We ran NC-ADMM, relax–round–polish, and the nuclear norm heuristic on each problem instance. For the nuclear norm heuristic, we solved the problem for 1000 values of λ and then polished the solution $\hat{\Sigma}^{\rm lr}$. In the polishing problem we replaced the rank constraint in problem (14) with the convex restriction $\Sigma^{\rm lr} \in \{Q_{1:k}\tilde{\Sigma}Q_{1:k}^{\rm T} \mid \tilde{\Sigma} \in \mathbf{S}^k_+\}$, where $\hat{\Sigma} = Q\Lambda Q^{\rm T}$ is the eigendecomposition of $\hat{\Sigma}^{\rm lr}$ and $Q_{1:k}$ is the first k columns of Q.

The $\Sigma^{\rm lr}$ and d values found by the three methods were always feasible solutions to problem (14). For each problem instance and each method, we took the value of the objective $\|\Sigma - \Sigma^{\rm lr} - D\|_F^2$ obtained by the method and subtracted the smallest objective value obtained by any method, $p^{\rm best}$. Figure 7 shows the average $\|\Sigma - \Sigma^{\rm lr} - D\|_F^2 - p^{\rm best}$ across all 24 months of returns data, for a given n and k. NC-ADMM always gave the best objective value on average (though not for each specific problem instance). The performance of relax–round–polish relative to NC-ADMM increased as k increased, while the relative performance of the nuclear norm heuristic decreased as k increased.

6.6 *Inexact graph isomorphism*

Two (undirected) graphs are isomorphic if we can permute the vertices of one so it is the same as the other (i.e. the same pairs of vertices are connected by edges). If we describe them by their adjacency matrices A and B, isomorphism is equivalent to the existence of a permutation matrix $Z \in \mathbf{R}^{n\times n}$ such that $ZAZ^{\mathrm{T}} = B$, or equivalently $ZA = BZ$.

Since in practical applications isomorphic graphs might be contaminated by noise, the inexact graph isomorphism problem is usually stated [1,17,84], in which we want to find a permutation matrix Z such that the disagreement $\|ZAZ^{\mathrm{T}} - B\|_F^2$ between the transformed matrix and the target matrix is minimized. Solving inexact graph isomorphism problems is of interest in pattern recognition [15,69], computer vision [71], shape analysis [39,73], image and video indexing [55], and neuroscience [86]. In many of the aforementioned fields graphs are used to represent geometric structures, and $\|ZAZ^{\mathrm{T}} - B\|_F^2$ can be interpreted as the strength of geometric deformation.

Since $\|ZAZ^{\mathrm{T}} - B\|_F^2 = \|ZA - BZ\|_F^2$ for any permutation matrix Z, the inexact graph isomorphism problem can be formulated as

$$\begin{array}{ll} \text{minimize} & \|ZA - BZ\|_F^2 \\ \text{subject to} & Z \in \mathcal{P}_n. \end{array} \tag{15}$$

If the optimal value of this problem is zero, it means that A and B are isomorphic. Otherwise, the solution of this problem minimizes the disagreement of ZAZ^{T} and B in the Frobenius norm sense. The non-convex set from Section 4 in problem (15) is the the set of permutation matrices $\mathcal{Z} = P_n$.

Problem instances. It can be shown that if A and B are isomorphic and A has distinct eigenvalues and all eigenvectors v of A satisfy $\mathbf{1}^{\mathrm{T}}v \neq 0$, then the relaxed problem has a unique solution which is the permutation matrix that relates A and B [1]. Hence in our first experiment, in order to generate harder problems, we generated the matrix A such that it violated these conditions. In particular, we constructed A for the Peterson graph (3-regular with 10 vertices), icosahedral graph (5-regular with 12 vertices), Ramsey graph (8-regular with 17 vertices), dodecahedral graph (3-regular with 20 vertices), and the Tutte-Coxeter graph (3-regular with 30 vertices). For each example we randomly permuted the vertices to obtain two isomorphic graphs.

We also used random graphs from the SIVALab data set [21] in our second experiment. These are Erdos–Renyi graphs that have been used for benchmarking different graph isomorphism algorithms. We ran our NCVX heuristic and Gurobi on 100 problems of size $n = 20$, 40,60,80.

Results. We implemented a faster version of NC-ADMM that caches work between convex solves. We ran our implementation with 25 iterations, 2 restarts, and no neighbour search. For all of our examples in the first experiment NC-ADMM was able to find the permutation relating the two graphs. It is interesting to notice that running the algorithm multiple times can find different solutions if there is more than one permutation relating the two graphs.

We compared NC-ADMM with Gurobi on random example in our second experiment. We ran Gurobi with a time limit of 300 seconds. Whenever Gurobi found a permutation matrix that gave an objective value of 0 it immediately returned the solution since the lower bound 0 was evident. NC-ADMM found a permutation solution for 97 out of 100 examples.

Figure 8 shows the runtime performance of the two methods. Each point shows how long NC-ADMM or Gurobi ran on a particular problem instance. Points with time component of 300 s indicate instances that Gurobi was unable to find a solution within the time limit. The goal of this comparison is to test the performance of generic methods on the graph isomorphism problem; tailored methods for this problem are significantly faster than both of these methods.

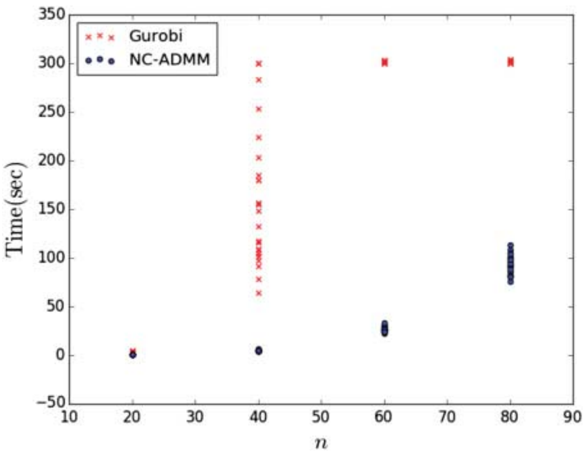

Figure 8. Time comparison of Gurobi and NC-ADMM on random graph isomorphism problems. Each point shows how long NC-ADMM or Gurobi ran on a particular problem instance.

7. Conclusion

We have discussed the relax–round–polish and NC-ADMM heuristics and demonstrated their performance on many different problems with convex objectives and decision variables from a non-convex set. Our heuristics are easy to extend to additional problems because they rely on a simple mathematical interface for non-convex sets. We need only know a method for (approximate) projection onto the set. We do not require but benefit from knowing a convex relaxation of the set, a convex restriction at any point in the set, and the neighbours of any point in the set under some discrete distance metric. Adapting our heuristics to any particular problem is straightforward, and we have fully automated the process in the NCVX package.

We do not claim that our heuristics give state-of-the-art results for any particular problem. Rather, the purpose of our heuristics is to give a fast and reasonable solution with minimal tuning for a wide variety of problems. Our heuristics also take advantage of the tremendous progress in technology for solving general convex optimization problems, which makes it practical to treat solving a convex problem as a black box.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. [DGE-114747] and by the DARPA XDATA and SIMPLEX programs.

References

[1] Y. Aflalo, A. Bronstein, and R. Kimmel, *On convex relaxation of graph isomorphism*, Proc. Natl. Acad. Sci. USA 112 (2015), pp. 2942–2947.

[2] W. Anderson and T. Morley, *Eigenvalues of the Laplacian of a graph*, Linear Multilinear Algebra 18 (1985), pp. 141–145.

[3] D. Applegate, R. Bixby, V. Chvatal, and W. Cook, *Concorde TSP solver*, preprint (2006). Available at http://www.math.uwaterloo.ca/tsp/concorde.html.

[4] N. Aybat, S. Zarmehri, and S. Kumara, *An ADMM algorithm for clustering partially observed networks*, Proceedings of the SIAM International Conference on Data Mining, 2015.
[5] D. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York, 2014.
[6] D. Boley, *Local linear convergence of the alternating direction method of multipliers on quadratic or linear programs*, SIAM J. Optim. 23 (2013), pp. 2183–2207.
[7] S. Boyd, M. Hast, and K. Astrom, *MIMO PID tuning via iterated LMI restriction*, Internat. J. Robust Nonlinear Control 26 (2015), pp. 1718–1731.
[8] 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. 3 (2011), pp. 1–122.
[9] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, 2004.
[10] P. Brucker, B. Jurisch, and B. Sievers, *A branch and bound algorithm for the job-shop scheduling problem*, Discrete Appl. Math. 49 (1994), pp. 107–127.
[11] I. Castillo, F. Kampas, and J. Pintér, *Solving circle packing problems by global optimization: Numerical results and industrial applications*, European J. Oper. Res. 191 (2008), pp. 786–802.
[12] R. Chartrand, *Nonconvex splitting for regularized low-rank + sparse decomposition*, IEEE Trans. Signal Process. 60 (2012), pp. 5810–5819.
[13] R. Chartrand and B. Wohlberg, *A nonconvex ADMM algorithm for group sparsity with sparse groups*, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 6009–6013.
[14] C. Collins and K. Stephenson, *A circle packing algorithm*, Comput. Geom. 25 (2003), pp. 233–256.
[15] D. Conte, P. Foggia, C. Sansone, and M. Vento, *Thirty years of graph matching in pattern recognition*, Int. J. Pattern Recognit. Artif. Intell. 18 (2004), pp. 265–298.
[16] J. Crawford and L. Auton, *Experimental results on the crossover point in random 3-SAT*, Artif. Intell. 81 (1996), pp. 31–57.
[17] A. Cross, R. Wilson, and E. Hancock, *Inexact graph matching using genetic search*, Pattern Recognit. 30 (1997), pp. 953–970.
[18] G. Dantzig, R. Fulkerson, and S. Johnson, *Solution of a large-scale traveling-salesman problem*, J. Oper. Res. Soc. Am. 2 (1954), pp. 393–410.
[19] J. David and B. De Moor, *The opposite of analytic centering for solving minimum rank problems in control and identification*, Proceedings of the IEEE Conference on Decision and Control, 1993, pp. 2901–2902.
[20] N. Derbinsky, J. Bento, V. Elser, and J. Yedidia, *An improved three-weight message-passing algorithm*, preprint (2013), arXiv preprint arXiv:1305.1961.
[21] M. De Santo, P. Foggia, C. Sansone, and M. Vento, *A large database of graphs and its use for benchmarking graph isomorphism algorithms*, Pattern Recognit. Lett. 24 (2003), pp. 1067–1079.
[22] S. Diamond and S. Boyd, *CVXPY: A Python-embedded modeling language for convex optimization*, J. Mach. Learn. Res. 17 (2016), pp. 1–5.
[23] G. Di Pillo and L. Grippo, *Exact penalty functions in constrained optimization*, SIAM J. Control Optim. 27 (1989), pp. 1333–1360.
[24] J. Eckstein and D. Bertsekas, *On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Program. 55 (1992), pp. 293–318.
[25] J. Eckstein and W. Yao, *Understanding the convergence of the alternating direction method of multipliers: Theoretical and computational perspectives*, Pac. J. Optim. 11 (2015), pp. 619–644.
[26] T. Erseghe, *Distributed optimal power flow using ADMM*, IEEE Trans. Power Syst. 29 (2014), pp. 2370–2380.
[27] R. Fletcher, *An exact penalty function for nonlinear programming with inequalities*, Math. Program. 5 (1973), pp. 129–150.
[28] A. Frank and A. Asuncion, *University of California, Irvine machine learning repository*, preprint (2010). Available at http://archive.ics.uci.edu/ml.
[29] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning*, Vol. 1, Springer, New York, 2001.
[30] D. Gabay and B. Mercier, *A dual algorithm for the solution of nonlinear variational problems via finite element approximation*, Comput. Math. Appl. 2 (1976), pp. 17–40.
[31] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, *Optimal parameter selection for the alternating direction method of multipliers (ADMM): Quadratic problems*, IEEE Trans. Automat. Control 60 (2015), pp. 644–658.
[32] P. Giselsson and S. Boyd, *Diagonal scaling in Douglas–Rachford splitting and ADMM*, Proceedings of the IEEE Conference on Decision and Control, 2014, pp. 5033–5039.
[33] P. Giselsson and S. Boyd, *Monotonicity and restart in fast gradient methods*, Proceedings of the IEEE Conference on Decision and Control, 2014, pp. 5058–5063.
[34] P. Giselsson and S. Boyd, *Preconditioning in fast dual gradient methods*, Proceedings of the IEEE Conference on Decision and Control, 2014, pp. 5040–5045.
[35] R. Glowinski and A. Marroco, *Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires*, Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique 9 (1975), pp. 41–76.
[36] M. Goldberg, *The packing of equal circles in a square*, Math. Mag. 43 (1970), pp. 24–30.
[37] Gurobi Optimization, Inc., *Gurobi Optimizer Reference Manual*, preprint (2015). Available at http://www.gurobi.com.
[38] S. Han and O. Mangasarian, *Exact penalty functions in nonlinear programming*, Math. Program. 17 (1979), pp. 251–269.

[39] L. He, C. Han, and W. Wee, *Object recognition and recovery by skeleton graph matching*, Proceedings of the IEEE International Conference on Multimedia and Expo, 2006, pp. 993–996.
[40] M. Hestenes, *Multiplier and gradient methods*, J. Optim. Theory Appl. 4 (1969), pp. 303–320.
[41] M. Hifi and R. M'Hallah, *A literature review on circle and sphere packing problems: Models and methodologies*, Adv. Oper. Res. 2009 (2009), pp. 1–22.
[42] H. Hmam, *Quadratic optimization with one quadratic equality constraint*, Tech. Rep., Electronic Warfare and Radar Division, Defence Science and Technology Organisation (DSTO), Australia, 2010.
[43] K. Hoffman, M. Padberg, and G. Rinaldi, *Traveling salesman problem*, in *Encyclopedia of Operations Research and Management Science*, Saul I. Gass and Michael C. Fu, eds., Springer, New York, 2013, pp. 1573–1578.
[44] M. Hong, *A distributed, asynchronous and incremental algorithm for nonconvex optimization: An ADMM approach*, preprint (2014). Available at https://arxiv.org/abs/1412.6058.
[45] M. Hong and Z. Luo, *On the linear convergence of the alternating direction method of multipliers*, Math. Program. 162 (2017), pp. 165–199.
[46] M. Hong, Z. Luo, and M. Razaviyayn, *Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems*, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2015, pp. 3836–3840.
[47] H. Hoos, *SATLIB — benchmark problems*, preprint (2016). Available at http://www.cs.ubc.ca/ ∼ hoos/SATLIB/benchm.html.
[48] K. Huang and N. Sidiropoulos, *Consensus-ADMM for general quadratically constrained quadratic programming*, IEEE Trans. Signal Process. 64 (2016), pp. 5297–5310.
[49] B. Jiang, S. Ma, and S. Zhang, *Alternating direction method of multipliers for real and complex polynomial optimization models*, Optimization 63 (2014), pp. 883–898.
[50] R. Kalman, *Identification of noisy systems*, Russian Math. Surveys 40 (1985), pp. 25–42.
[51] J. Kruskal, *On the shortest spanning subtree of a graph and the traveling salesman problem*, Proc. Am. Math. Soc. 7 (1956), pp. 48–50.
[52] H. Kuhn, *The Hungarian method for the assignment problem*, Nav. Res. Logist. 52 (2005), pp. 7–21.
[53] E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, Chichester, 1985.
[54] E. Lawler and D. Wood, *Branch-and-bound methods: A survey*, Oper. Res. 14 (1966), pp. 699–719.
[55] J. Lee, *A graph-based approach for modeling and indexing video data*, Proceedings of the IEEE International Symposium on Multimedia, 2006, pp. 348–355.
[56] G. Li and T. Pong, *Global convergence of splitting methods for nonconvex composite optimization*, SIAM J. Optim. 25 (2015), pp. 2434–2460.
[57] A. Liavas and N. Sidiropoulos, *Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers*, IEEE Trans. Signal Process. 63 (2015), pp. 5450–5463.
[58] T. Lipp and S. Boyd, *Variations and extension of the convex–concave procedure*, Optim. Eng. (2014), pp. 1–25.
[59] R. Merris, *Laplacian matrices of graphs: A survey*, Linear Algebra Appl. 197 (1994), pp. 143–176.
[60] D. Mitchell, B. Selman, and H. Levesque, *Hard and easy distributions of SAT problems*, Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 92, 1992, pp. 459–465.
[61] J. Mota, J. Xavier, P. Aguiar, and M. Püschel, *Basis pursuit in sensor networks*, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2011, pp. 2916–2919.
[62] P. Narendra and K. Fukunaga, *A branch and bound algorithm for feature subset selection*, IEEE Trans. Comput. 100 (1977), pp. 917–922.
[63] L. Ning, T. Georgiou, T. Tryphon, A. Tannenbaum, and S. Boyd, *Linear models based on noisy data and the Frisch scheme*, SIAM Rev. 57 (2015), pp. 167–197.
[64] M. Padberg and G. Rinaldi, *A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems*, SIAM Rev. 33 (1991), pp. 60–100.
[65] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Dover, Mineola, NY, 1998.
[66] Z. Peng, J. Chen, and W. Zhu, *A proximal alternating direction method of multipliers for a minimization problem with nonconvex constraints*, J. Global Optim. (2015), pp. 1–18.
[67] A. Perold, *Large-scale portfolio optimization*, Manag. Sci. 30 (1984), pp. 1143–1160.
[68] M. Powell, *Algorithms for nonlinear constraints that use Lagrangian functions*, Math. Program. 14 (1978), pp. 224–248.
[69] J. Rocha and T. Pavlidis, *A shape analysis model with applications to a character recognition system*, IEEE Trans. Pattern Anal. Mach. Intell. 16 (1994), pp. 393–404.
[70] J. Saunderson, V. Chandrasekaran, P. Parrilo, and A. Willsky, *Diagonal and low-rank matrix decompositions, correlation matrices, and ellipsoid fitting*, SIAM J. Matrix Anal. Appl. 33 (2012), pp. 1395–1416.
[71] C. Schellewald, S. Roth, and C. Schnörr, *Evaluation of convex optimization techniques for the weighted graph-matching problem in computer vision*, in *Pattern Recognition*, B. Radig and S. Florczyk, eds., Springer, Berlin, 2001, pp. 361–368.
[72] L. Schizas, A. Ribeiro, and G. Giannakis, *Consensus in ad hoc WSNs with noisy links—part I: Distributed estimation of deterministic signals*, IEEE Trans. Signal Process. 56 (2008), pp. 350–364.
[73] T. Sebastian, P. Klein, and B. Kimia, *Recognition of shapes by editing their shock graphs*, IEEE Trans. Pattern Anal. Mach. Intell. 26 (2004), pp. 550–571.

[74] H. Sedghi, A. Anandkumar, and E. Jonckheere, *Multi-step stochastic ADMM in high dimensions: Applications to sparse optimization and matrix decomposition*, in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, Curran Associates, Red Hook, NY, 2014, pp. 2771–2779.
[75] W. Sharpe, *Portfolio Theory and Capital Markets*, McGraw-Hill, New York, 1970.
[76] H. Sherali and W. Adams, *A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems*, SIAM J. Discrete Math. 3 (1990), pp. 411–430.
[77] E. Specht, *Packomania*, preprint (2013). Available at http://www.packomania.com/.
[78] J. Spingarn, *Applications of the method of partial inverses to convex programming: Decomposition*, Math. Program. 32 (1985), pp. 199–223.
[79] K. Stephenson, *Introduction to Circle Packing: The Theory of Discrete Analytic Functions*, Cambridge University Press, Cambridge, 2005.
[80] R. Stubbs and S. Mehrotra, *A branch-and-cut method for 0-1 mixed convex programming*, Math. Program. 86 (1999), pp. 515–532.
[81] R. Takapoui, N. Moehle, S. Boyd, and A. Bemporad, *A simple effective heuristic for embedded mixed-integer quadratic programming*, Proceedings of the American Control Conference, 2016, pp. 5620–5625.
[82] M. Tawarmalani and N. Sahinidis, *A polyhedral branch-and-cut approach to global optimization*, Math. Program. 103 (2005), pp. 225–249.
[83] R. Tibshirani, *Lecture notes in modern regression*, preprint (2013). Available at http://www.stat.cmu.edu/ ∼ ryantibs/datamining/lectures/17-modr 2.pdf.
[84] S. Umeyama, *An eigendecomposition approach to weighted graph matching problems*, IEEE Trans. Pattern Anal. Mach. Intell. 10 (1988), pp. 695–703.
[85] L. Vandenberghe and S. Boyd, *Semidefinite programming*, SIAM Rev. 38 (1996), pp. 49–95.
[86] J. Vogelstein, J. Conroy, L. Podrazik, S. Kratzer, E. Harley, D. Fishkind, R. Vogelstein, and C. Priebe, *Fast approximate quadratic programming for graph matching*, PLOS ONE 10 (2015), pp. 1–17.
[87] J. Von Neumann, *Some matrix inequalities and metrization of metric space*, Tomsk Univ. Rev. 1 (1937), pp. 286–296.
[88] B. Wahlberg, S. Boyd, M. Annergren, and Y. Wang, *An ADMM algorithm for a class of total variation regularized estimation problems*, IFAC Proc. Vol. 45 (2012), pp. 83–88.
[89] D. Wang, H. Lu, and M. Yang, *Online object tracking with sparse prototypes*, IEEE Trans. Image Process. 22 (2013), pp. 314–325.
[90] F. Wang, Z. Xu, and H. Xu, *Convergence of Bregman alternating direction method with multipliers for nonconvex composite problems*, preprint (2014), arXiv preprint arXiv:1410.8625v3.
[91] Y. Xu, W. Yin, Z. Wen, and Y. Zhang, *An alternating direction algorithm for matrix completion with nonnegative factors*, Front. Math Sci. China 7 (2012), pp. 365–384.
[92] R. Zhang and J. Kwok, *Asynchronous distributed ADMM for consensus optimization*, Proceedings of the International Conference on Machine Learning, 2014, pp. 1701–1709.