Variations and Extensions of the Convex-Concave Procedure

Thomas Lipp and Stephen Boyd

August 7, 2014

Abstract

We investigate the convex-concave procedure (CCP), a local heuristic that utilizes the tools of convex optimization to find local optima of difference of convex (DC) programming problems. The class of DC problems includes many difficult problems such as the traveling salesman problem. We extend the standard procedure in two major ways and describe several variations. First, we allow for the algorithm to be initialized without a feasible point. Second, we generalize the algorithm to include vector inequalities. We then present several examples to demonstrate these algorithms.

1 Introduction

In this paper we present several extensions of and variations on the convex-concave procedure (CCP), a powerful heuristic method used to find local solutions to difference of convex (DC) programming problems. We then demonstrate the algorithms with several examples.

1.1 Difference of convex programming

In this paper we consider DC programming problems, which have the form

$$\begin{array}{ll} \text{minimize} & f_0(x) - g_0(x) \\ \text{subject to} & f_i(x) - g_i(x) \leq 0, \quad i = 1, \dots, m, \end{array} \tag{1}$$

where $x \in \mathbf{R}^n$ is the optimization variable and $f_i : \mathbf{R}^n \to \mathbf{R}$ and $g_i : \mathbf{R}^n \to \mathbf{R}$ for $i = 0, \dots, m$ are convex. The class of DC functions is very broad; for example, any C^2 function can be expressed as a difference of convex functions [Har59]. A DC program is not convex unless the functions g_i are affine, and is hard to solve in general. To see this, we can cast the Boolean linear program (LP)

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & x_i \in \{0, 1\}, \quad i = 1, \dots, n \\ & Ax \leq b, \end{array} \tag{2}$$

where $x \in \mathbf{R}^n$ is the optimization variable and $c \in \mathbf{R}^n$, $A \in \mathbf{R}^{m\times n}$, and $b \in \mathbf{R}^m$ are problem data, in the DC form (1) as

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & x_i^2 - x_i \leq 0, \quad i = 1, \ldots, n \\ & x_i - x_i^2 \leq 0, \quad i = 1, \ldots, n \\ & Ax - b \leq 0. \end{array} \tag{3}$$

Here the objective and constraint functions are convex, except for the second block of n inequality constraint functions, which are concave. Thus the Boolean LP (2) is a subclass of DC programs (1). The Boolean LP, in turn, can represent many problems that are thought to be hard to solve, like the traveling salesman problem; for these problems, no polynomial time algorithm is known, and it is widely believed none exists [Kar72]. We will examine one instance from this class, 3-satisfiability, in §5.

The global solution to (1) can be found through general branch and bound methods [Agi66, LW66]. There is also an extensive literature on solving DC programming problems which we will review later. Alternatively, one can attempt to find a local optimum to this problem through the many techniques of nonlinear optimization [NW06].

1.2 Convex-concave procedure

We now present the basic convex-concave procedure, also known as the concave-convex procedure [YR03]. This is one heuristic for finding a local optimum of (1) that leverages the ability to efficiently solve convex optimization problems.

We will assume that all of the f_i and g_i are differentiable for the ease of notation, but the analysis holds for nondifferentiable functions where the gradient at a point is replaced by a subgradient at that point. This basic version of the algorithm requires an initial feasible point x_0, *i.e.*, $f_i(x_0) - g_i(x_0) \leq 0$, for $i = 1, \ldots, m$.

Algorithm 1.1 *Basic CCP algorithm.*

```
given an initial feasible point $x_0$.
$k := 0$.
repeat
    1. Convexify. Form $\hat{g}_i(x; x_k) \triangleq g_i(x_k) + \nabla g_i(x_k)^T (x - x_k)$ for $i = 0, \ldots, m$.
    2. Solve. Set the value of $x_{k+1}$ to a solution of the convex problem
           minimize    $f_0(x) - \hat{g}_0(x; x_k)$
           subject to  $f_i(x) - \hat{g}_i(x; x_k) \leq 0, \quad i = 1, \ldots, m$.
    3. Update iteration. $k := k + 1$.
until stopping criterion is satisfied.
```

One reasonable stopping criterion is that the improvement in the objective value is less than some threshold δ, *i.e.*,

$$(f_0(x_k) - g_0(x_k)) - (f_0(x_{k+1}) - g_0(x_{k+1})) \leq \delta.$$

We will see the lefthand side is always nonnegative. Observe that the subproblem in step 2 of algorithm 1.1,

$$\begin{array}{ll} \text{minimize} & f_0(x) - \big(g_0(x_k) + \nabla g_i(x_k)^T(x - x_k)\big) \\ \text{subject to} & f_i(x) - \big(g_i(x_k) + \nabla g_i(x_k)^T(x - x_k)\big) \leq 0, \quad i = 1, \dots, m \end{array} \tag{4}$$

is convex, since the objective and constraint functions are convex, and can therefore be solved efficiently.

Initialization. CCP is a local heuristic, and thus, the final point found may (and often does) depend on the initial point x_0. It is therefore typical to initialize the algorithm with several (feasible) x_0 and take as the the final choice of x the final point found with the lowest objective value over the different runs. The initial point x_0 can be chosen randomly (provided that feasibility is ensured) or through a heuristic, if one is known.

Line search. Unlike some algorithms, CCP does not require a line search. However, a line search may still be performed. In particular taking a larger step can lead to faster convergence.

Algorithm 1.2 *Line search for CCP.*

```
given a solution $x_{k+1}$ to (4) and $\alpha > 1$.
$t := 1$.
while $f_0(x_k + \alpha t(x_{k+1} - x_k)) - g_0(x_k + \alpha t(x_{k+1} - x_k)) \leq f_0(x_k) - g_0(x_k)$ and
    $f_i(x_k + \alpha t(x_{k+1} - x_k)) - g_i(x_k + \alpha t(x_{k+1} - x_k)) \leq 0$, for $i = 1, \dots, m$,
    $t := \alpha t$.
$x_{k+1} := x_k + t(x_{k+1} - x_k)$.
```

1.3 Convergence proof

We will first observe that all of the iterates are feasible, and then show that CCP is a descent algorithm, *i.e.*,

$$f_0(x_{k+1}) - g_0(x_{k+1}) \leq f_0(x_k) - g_0(x_k).$$

Assume x_k is a feasible point for (1). We know that x_k is a feasible point for the convexified subproblem (4) because

$$f_i(x_k) - g_i(x_k; x_k) = f_i(x_k) - g_i(x_k) \leq 0,$$

so a feasible point x_{k+1} exists to the convexified subproblem (4). The convexity of g_i gives us $\hat{g}_i(x; x_k) \leq g_i(x)$, for all x, so

$$f_i(x) - g_i(x) \leq f_i(x) - \hat{g}_i(x; x_k).$$

It then follows that x_{k+1} must be a feasible point of (1) since

$$f_i(x_{k+1}) - g_i(x_{k+1}) \leq f_i(x_{k+1}) - \hat{g}_i(x_{k+1}; x_k) \leq 0.$$

Thus, because x_0 was chosen feasible, all iterates are feasible.

We will now show that the objective value converges. Let $v_k = f_0(x_k) - g_0(x_k)$. Then

$$v_k = f_0(x_k) - g_0(x_k) = f_0(x_k) - \hat{g}_0(x_k; x_k) \geq f_0(x_{k+1}) - \hat{g}_0(x_{k+1}; x_k),$$

where the last inequality follows because at each iteration k we minimize the value of $f_0(x) - \hat{g}_0(x; x_k)$, and we know that we can achieve v_k by choosing $x_{k+1} = x_k$. Thus

$$v_k \geq f_0(x_{k+1}) - \hat{g}_0(x_{k+1}; x_k) \geq v_{k+1}.$$

Thus the sequence $\{v_i\}_{i=0}^{\infty}$ is nonincreasing and will converge, possible to negative infinity. A proof showing convergence to critical points of the original problem can be found in [LS09].

Although the objective value converges, it does not necessarily converge to a local minimum. Consider the problem

$$\text{minimize} \quad x^4 - x^2,$$

where $x \in \mathbf{R}$ is the optimization variable, which has optimal value -0.25 at $x = \pm 1/\sqrt{2}$. If the algorithm is initialized with $x_0 = 0$, then the algorithm will converge in one step to the local maximum value, 0.

1.4 Advantages of convex-concave procedure

One of the advantages of CCP over other algorithms, like sequential quadratic programming (SQP), is that more information is retained in each of the iterates. In SQP the problem at each iteration is approximated by a quadratic program (convex quadratic objective and linear constraints). Thus all information above the second order is lost in the objective and even more is lost in the constraints. On the other hand, CCP is able to retain all of the information from the convex component of each term and only linearizes the concave portion.

Another advantage of CCP is that the over estimators $f_i(x) - \hat{g}_i(x; x_k)$ are global. Many approximation procedures, like SQP, require trust regions which limit progress at an iteration to a region where the approximation is valid [BGN00]. Because of the global nature of the inequalities for convex and concave functions, our bounds are valid everywhere. We therefore do not need to limit the progress at each step or perform a line search. Although SQP and other approximation algorithms were popular for the ease of solving each step, as the technology for solving more general convex programs has improved it has become beneficial to take advantage of greater information.

1.5 Outline

In §2 we examine previous work in solving DC programs and the history of iterative convexification procedures. In §3.1 we will introduce our first extension to CCP in which constraints are loosened. In §4 we present our second extension of CCP, giving a vector version of the algorithm, which is particularly relevant for matrix constraints. In §5 we present several examples using these methods including 3-satisfiability, circle packing, circuit layout, and multi-matrix principal component analysis.

2 Previous work

Difference of convex programming problems of the form (1) have been studied for several decades. Early approaches to solving the problem globally often involved transforming the problem into a concave minimization problem (minimize a concave function over a convex set) or a reverse convex problem (a convex optimization problem except for a constraint of the form $f(x) > 0$ where f is convex) [Tuy86, TH88, HPTDV91]. Good overviews of the work that has been done in solving DC programs globally can be found in [HPT95, HT99], and the references therein.

Solving DC problems globally, and the related concave minimization and reverse convex optimization problems, most often rely on branch and bound or cutting plane methods as in [MF97]. Branch and bound methods were originally popularized for combinatorial problems [Agi66, LW66] but soon made the transition to general nonconvex optimization [FS69, Sol71, Hor86]. Branch and bound methods involve splitting the domain into partitions on which simpler problems can be solved to find upper and lower bounds on the optimal value in that region. Further subdividing these regions will produce tighter bounds. The hope is that these bounds will eliminate regions so that exploration of the entire domain will prove unnecessary.

The subproblems created by branch and bound methods are often reverse convex problems, a term first coined in [Mey70], or concave minimization problems. These problems are often approached with simplicial algorithms as in [Hil75, HJ80a] or cutting plane methods as in [HJ80b, TT80, Muu85]. Cutting plane methods, an early optimization technique as seen in [Zan69], involve adding constraints that eliminate regions of the domain known not to contain the solution. Another popular approach for addressing the concave minimization problem is outer approximation as discussed in [FH76, Tuy83, HTB91] and the less common inner approximation as in [YTI00]. A more in depth discussion of these problem classes and approaches can be found in [HT96].

However, these global methods often prove slow in practice, requiring many partitions or cuts. Therefore, we are instead concerned with local heuristics that can find improved solutions rapidly. The sequential nature of CCP draws from the tradition of sequential quadratic programming (SQP). SQP was introduced in [Wil63] with convergence properties shown in [Rob72]. SQP typically involves approximating an optimization problem by a quadratic objective with linear constraints. This approximation is then used to find a search direction for descent of the original problem. SQP is a well developed field and much more can be learned about the process from [BT95, GW12, NW06] and the references therein.

CCP can also be considered a generalization of majorization minimization (MM) algorithms, of which expectation maximization (EM) is the most famous. Expectation maximization was introduced in [DLR77] and although MM algorithms are just as old, as seen in [DL77], the term majorization minimization was not coined until Hunter and Lange's rejoinder to [LHY00]. In MM algorithms, a difficult minimization problem is approximated by an easier to minimize upper bound created around a particular point, a step called majorization. The minimum of that upper bound (the minimization step) is then used to sequentially create another, hopefully tighter, upper bound (another majorization step) to be minimized. Although the name may be new, many algorithms, including gradient and Newton methods, may be thought of as MM schemes. Many EM and MM extensions have been developed over the years and more can be found on these algorithms in [LR87, Lan04, MK07]. There have even been approaches that have started to address the DC programming problem as in [LHY00].

Although many algorithms reduce to CCP, including EM, it is not the only time that sequential convex optimization algorithms have been created. In the field of structural optimization, an algorithm called sequential convex optimization is used. First introduced as the method of moving asymptotes in [Sva87] and later expanded in [Zil01] this method similarly involves sequential convexification, although the parameters are rescaled to drive solutions away from variable limits. In vehicle avoidance and trajectory problems, many have independently developed their own sequential convexification procedures which are effectively CCP procedures. A few examples include [ML08] and [SLA$^+$13].

The convex-concave procedure was first introduced geometrically in [YR03] although without inequality constraints. Our approach and analysis more closely follows that in [SVH05] which considered the procedure as a sequence of convex optimization problems and added inequality constraints; algorithm 1.1 is almost identical to their "Constrained Concave Convex Procedure". CCP is already used in a variety of settings including image reconstruction as in [Byr00], Support Vector Machines (SVM) with additional structure as in [YJ09], and even for tuning multi-input multi-output proportional-integral-derivative control as in [BHÅ14] which includes matrix constraints. In extending CCP we draw from techniques developed in many of its predecessors.

3 Penalty convex-concave

3.1 Basic algorithm

In this section we present our first extension to CCP, which removes the need for an initial feasible point. We relax our problem by adding slack variables to our constraints and penalizing the sum of the violations. It is well known in SQP and other iterative techniques that the individual iterates may not be feasible, prompting the use of slack variables [GPM76, Pow78]. Here, rather than using slack variables as a quick fix, we leverage them in our algorithm. By initially putting a low penalty on violations, we allow for constraints to be violated so that a region with lower objective value can be found. Thus this approach may be desirable even if a feasible initial point is known. This approach can similarly be thought of as modeling our constraints with a hinge loss. Penalizing the sum of violations is equivalent to using the

ℓ_1 norm and is well known to induce sparsity [BV04, §6.3.2]. Therefore, if we are unable to satisfy all constraints, the set of violated constraints should be small.

Algorithm 3.1 *Penalty CCP.*

```
given an initial point $x_0$, $\tau_0 > 0$, $\tau_{\max}$, and $\mu > 1$.
$k := 0$.
repeat
    1. Convexify. Form $\hat{g}_i(x; x_k) \triangleq g_i(x_k) + \nabla g_i(x_k)^T(x - x_k)$ for $i = 0, \ldots, m$.
    2. Solve. Set the value of $x_{k+1}$ to a solution of
         minimize    $f_0(x) - \hat{g}_0(x; x_k) + \tau_k \sum_{i=1}^m s_i$
         subject to  $f_i(x) - \hat{g}_i(x; x_k) \le s_i, \quad i = 1, \ldots, m$
                     $s_i \ge 0, \quad i = 1, \ldots, m.$
    3. Update $\tau$. $\tau_{k+1} := \min(\mu\tau_k, \tau_{\max})$.
    4. Update iteration. $k := k + 1$.
until stopping criterion is satisfied.
```

One reasonable stopping criteria would be when the improvement in objective when solving the convexified problem is small, *i.e.*,

$$\left(f_0(x_k) - g_0(x_k) + \tau_k \sum_{i=1}^m s_i^k\right) - \left(f_0(x_{k+1}) - g_0(x_{k+1}) + \tau_k \sum_{i=1}^m s_i^{(k+1)}\right) \le \delta,$$

where s_i^k is the slack variable s_i found at iteration k, and either x_k is feasible, *i.e.*,

$$\sum_{i=1}^m s_i^{(k+1)} \le \delta_{\text{violation}} \approx 0,$$

or $\tau_k = \tau_{\max}$.

This algorithm is not a descent algorithm, but the objective value will converge, although the convergence may not be to a feasible point of the original problem. To see this convergence observe that once $\tau = \tau_{\max}$, we can rewrite the problem with slack variables as (1) and then algorithm 3.1 is equivalent to algorithm 1.1 and the objective will therefore converge.

The upper limit $\tau_{\max}$ on τ is imposed to avoid numerical problems if τ_i grows too large and to provide convergence if a feasible region is not found. The theory of exact penalty functions tells us that if τ_i is greater than the largest optimal dual variable associated with the inequalities in the convexified subproblem (4), then solutions to (4) are solutions of the relaxed convexified problem, and subject to some conditions on the constraints, if a feasible point exists, solutions to the relaxed problem are solutions to the convexified problem (*e.g.*, $\sum_{i=1}^m s_i = 0$) [HM79, DPG89]. Provided $\tau_{\max}$ is larger than the largest optimal dual variable in the unrelaxed subproblems (4) the value of $\tau_{\max}$ will have no impact on the solution. This value is unlikely to be known; we therefore choose $\tau_{\max}$ large. Observe that, for sufficiently large $\tau_{\max}$, if (1) is convex, and the constraint conditions are met, then penalty CCP is not a heuristic but will find an optimal solution.

In the case of nondifferentiable functions, we do not specify a particular subgradient. However, the choice of subgradient can have an impact on the performance of the algorithm. We will see an example of this in §5.3.

3.2 Enforcing constraints

There are many modification possible to this algorithm in the way constraints are handled. For example, the value of τ could be chosen on a per constraint basis, prioritizing the satisfaction of certain constraints over others.

Another variation is that constraints that are purely convex ($g_i(x) = 0$) could be enforced at all iterations without a slack variable. If a feasible point exists to the problem then clearly it must obey all of the convex constraints, so a feasible point will be found at each iteration without slack for the convex constraints. In the standard algorithm slack variables are included for convex constraints because temporarily violating a convex constraint may allow the algorithm, on subsequent iterations, to reach a more favorable region of a nonconvex constraint. Enforcing the constraints without slack, on the other hand, reduces the search area for the solution, and may lead to faster convergence and greater numerical stability.

Yet another variation is that once a constraint becomes satisfied, it could be handled as in algorithm 1.1, without a slack variable, guaranteeing that the constraint will be satisfied for all future iterates. It has been our experience in numerical implementation, that this last variation is ill advised and may constrain the algorithm prematurely to a region without feasible points

3.3 Cutting plane techniques

Often DC programs may have large numbers of constraints, as we will see in the circle packing problem of §5.2. However, most of these constraints are inactive ($f_i(x) - g_i(x) \ll 0$). In these instances, cutting plane or column generation techniques can be used [EM75, dMVDH99, MB09]. These methods keep track of a set of active and likely to become active constraints to include at each iteration while ignoring well satisfied constraints. There are many ways to choose these active constraints with various convergence properties; we describe two basic approaches here.

In the first approach, one can include every constraint that has been violated at any iteration of the algorithm. In the worst case scenario, this could result in all of the constraints being included, but is guaranteed to converge. In the second approach the n constraints with the largest value ($f_i(x) - g_i(x)$) are included at each iteration. This approach does not guarantee convergence, but for appropriate n works very well in many situations. By greatly reducing the number of constraints that need to be considered, much larger problems can be handled.

Although they may seem different, these methods derive from the same source as the cutting plane methods mentioned in §2 for solving concave minimization problems [Kel60].

The difference in this implementation is that when adding a constraint (cutting plane), it is simply chosen from the list of constraints in the original problem statement, in the methods in §2, new constraints need to be generated.

4 Vector convex-concave

We now generalize the DC problem (1) to include vector inequalities. Our problem statement is now

$$\begin{array}{ll} \text{minimize} & f_0(x) - g_0(x) \\ \text{subject to} & f_i(x) - g_i(x) \preceq_K 0, \quad i = 1, \ldots, m, \end{array} \tag{5}$$

where $x \in \mathbf{R}^n$ is the optimization variable, $f_0 : \mathbf{R}^n \to \mathbf{R}$ and $g_0 : \mathbf{R}^n \to \mathbf{R}$ are convex, $K \subseteq \mathbf{R}^p$ is a proper cone, and $f_i : \mathbf{R}^n \to \mathbf{R}^p$ and $g_i : \mathbf{R}^n \to \mathbf{R}^p$ for $i = 1, \ldots, m$ are K-convex. We use $\preceq_K$ to represent generalized inequalities with respect to the cone K, *e.g.*, $x \preceq_K y$ means $y - x \in K$, and $x \prec_K y$ means $y - x \in \textbf{int}\, K$. For more background on generalized inequalities and proper cones see [BV04, §2.4.1]. For more on convexity with respect to generalized inequalities see [BV04, §3.6.2]

We can now construct a generalization of algorithm 1.1 for the vector case. Again we require x_0 feasible, *e.g.*, $f_i(x_0) - g_i(x_0) \preceq_K 0$ for $i = 1, \ldots, m$.

Algorithm 4.1 *Vector CCP.*

```
given an initial feasible point $x_0$.
$k := 0$.
repeat
    1. Convexify. Form $\hat{g}_i(x; x_k) \triangleq g_0(x_k) + \nabla g_0(x_k)^T (x - x_k)$ and
       $g_i(x; x_k) \triangleq g_i(x_k) + Dg_i(x_k)(x - x_k)$ for $i = 1, \ldots, m$.
    2. Solve. Set the value of $x_{k+1}$ to a solution of
           minimize     $f_0(x) - \hat{g}_0(x; x_k)$
           subject to   $f_i(x) - \hat{g}_i(x; x_k) \preceq_K 0, \quad i = 1, \ldots, m$.
    3. Update iteration. $k := k + 1$.
until stopping criterion is satisfied.
```

In the above algorithm we use $Dg_i(x_k)$ to represent the derivative or Jacobian matrix of g_i at x_k. The proof of feasible iterates and convergence is identical except for the substitution of the generalized inequalities for inequalities and the Jacobian for the gradient. For more on convex optimization problems with generalized inequalities see [BV04, §4.6, 11.6].

Similarly we can extend algorithm 3.1 for the vector case.

Algorithm 4.2 *Penalty vector CCP.*

```
given an initial point $x_0$, $t_0 \succ_{K^*} 0$, $\tau_{\max}$, and $\mu > 1$.
$k := 0$.
repeat
    1. Convexify. Form $\hat{g}_i(x; x_k) \triangleq g_0(x_k) + \nabla g_0(x_k)^T (x - x_k)$ and
        $g_i(x; x_k) \triangleq g_i(x_k) + Dg_i(x_k)(x - x_k)$ for $i = 1, \ldots, m$.
    2. Solve. Set the value of $x_{k+1}$ to a solution of
        minimize    $f_0(x) - \hat{g}_0(x; x_k) + \sum_{i=1}^m t_k^T s_i$
        subject to  $f_i(x) - \hat{g}_i(x; x_k) \preceq_K s_i, \quad i = 1, \ldots, m$
                    $s_i \succeq_K 0, \quad i = 1, \ldots, m.$
    3. Update t.
        if $\|\mu t_k\|_2 \le \tau_{\max}$
            $t_{k+1} := \mu t_k$,
        else
            $t_{k+1} := t_k$.
    4. Update iteration. $k := k + 1$.
until stopping criterion is satisfied.
```

In the above algorithm K^* represents the dual cone of K. Since K is a proper cone, K^* is a proper cone, and therefore has an interior. Furthermore, if $t_i \succ_{K^*} 0$ then $\mu t_i \succ_{K^*} 0$ so all $t_i \succ_{K^*} 0$ for $i \ge 0$. Note that because $s_i \succeq_K 0$, $t_k^T s_i \ge 0$ at all iterations. Observe that if $t_k^T s_i = 0$, then $t_k^T s_i \le 0$ and $-s_i \in K$, so $s_i \preceq_K 0$, and

$$f_i(x) - g_i(x) \preceq_K f_i(x) - \hat{g}_i(x; x_k) \preceq_K s_i \preceq_K 0,$$

so the inequality is satisfied. As in algorithm 3.1, our objective increases the cost of violating the inequality at each iterate, driving the violations or slacks s_i towards zero. For more information on dual cones and generalized inequalities see [BV04, §2.6].

As before, once $\|\mu t_k\|_2 > \tau_{\max}$, we can rewrite our problem as (5) and then algorithm 4.2 is equivalent to 4.1, and will therefore converge, although not necessarily to a feasible point of the original problem.

5 Examples

We now present several examples using these algorithms. Each of these examples comes from a large field where much time has been spent developing algorithms targeted at these applications. It is not our intention to beat these algorithms but rather to show that this general approach performs remarkably well in a variety of settings.

5.1 3-satisfiability

General description. Satisfiability problems ask if there exists an assignment of Boolean variables such that an expression evaluates to **true**. In 3-satisfiability (3-SAT) the expression is a conjunction of expressions with three disjunctions and possibly negations.

Mathematical description. We can represent the 3-SAT problem as a Boolean LP (2) where m is the number of expressions and the entries of A are given by

$$a_{ij} = \begin{cases} -1 & \text{if expression } i \text{ is satisfied by } x_j \textbf{ true} \\ 1 & \text{if expression } i \text{ is satisfied by } x_j \textbf{ false} \\ 0 & \text{otherwise,} \end{cases}$$

and the entries of b are given by

$$b_i = 2 - (\text{number of negated terms in expression } i).$$

In this case, there is no objective, since we are only looking for a feasible point; c can be the zero vector.

Initialization procedure. We present two initialization procedures for this algorithm. In the first procedure we initialize the entries of x_0 by drawing from the uniform distribution on $[0, 1]$. In the second procedure we choose x_0 to be an optimal value of x in a linear programming (LP) relaxation of the problem

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{n} |x_i - 0.5| \\ \text{subject to} & 0 \preceq x \preceq 1, \\ & Ax \preceq b, \end{array} \tag{6}$$

where x is the optimization variable, A and b are problem data, and the objective function is chosen to not bias assignments towards **true** or **false**.

Algorithm variation. To solve the problem we use the variant of algorithm 3.1 presented in 3.2 that enforces the convex constraints at each iteration with no slack. Note that when we run the algorithm there is no guarantee that the resulting values will be integers, so to evaluate the solution we round the values of x.

Problem instance. To demonstrate the algorithm we used randomly generated 3-SAT problems of varying sizes. For randomly generated 3-SAT problems as defined in [MSL92] there is a threshold around 4.25 expressions per variable when problems transition from being feasible to being infeasible [CA96]. Problems near this threshold are generally found to be hard satisfiability problems. We only test problems below this threshold, because the algorithm provides no certificate of infeasibility.

For each problem and constraint size below the feasibility threshold, 10 problem instances were created and the existence of a satisfiable assignment was verified using the integer programming solver MOSEK [MOS13]. In the case of random initialization, 10 initializations were tried for each problem and if any of them found an x that, when rounded, satisfied all expressions, success was reported.

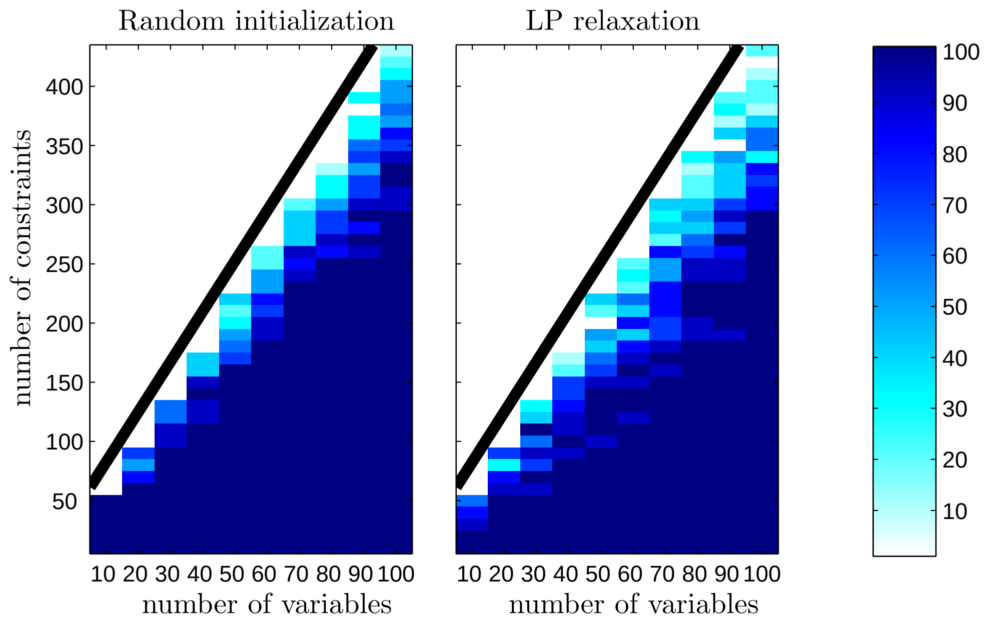

Figure 1: Percentage of runs for which a satisfying assignment to random 3-SAT problems were found for problems of varying sizes.

Computational details. The subproblems were solved using CVX [CR12, GB08] as the interface to the SDPT3 solver [TTT99, TTT03] on a 2.66 GHz Intel Core 2 Duo machine. For a problem with 100 variables and 430 expressions, the algorithm took between 5 and 25 steps, depending on the initialization with an average of 11 steps; the average solve time for each subproblem was under one second.

Results. Figure 1 compares the results of random initialization with the LP relaxation initialization (6). Using random initializations, satisfying assignments could be found consistently for up to 3.5 constraints per variable at which point success started to decrease.

Figure 2 depicts a histogram of the number of expressions not satisfied over 1000 random initializations for a problem with 100 variables and 430 expressions. Observe that in the linear inequality formulation of 3-SAT used, there is no constraint driving the number of violations to be sparse. When the convex constraints are enforced, the objective encourages fewer variables to be noninteger valued, rather than fewer expressions to be unsatisfied.

5.2 Circle packing

General description. The circle packing problem finds the largest percentage of a polygon that can be covered by a set number of circles of equal radius. This problem has long been studied in mathematics, and databases exist of the densest known packings for different numbers of circles in a square [Spe13].

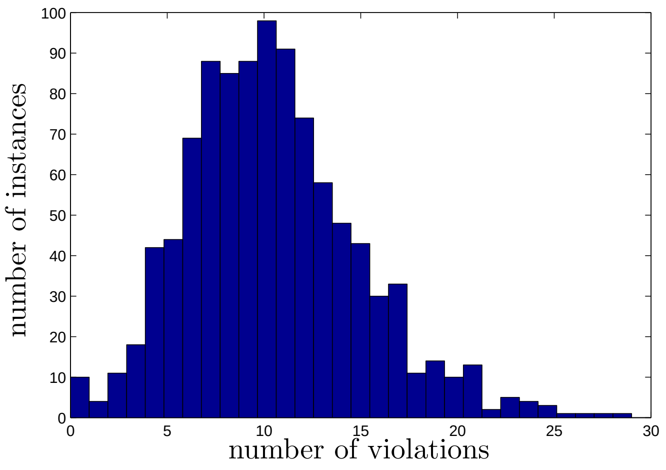

Figure 2: Histogram of the number of unsatisfied expressions for a hard 3-SAT problem instance with 100 variables and 430 expressions.

Mathematical description. For our example we will consider n circles and take the polygon to be a square with side length l. Let $x_i \in \mathbf{R}^2$ for $i = 1, \ldots, n$ represent the position of circle i and r be radius of the circles. The problem is

$$\begin{array}{ll} \text{maximize} & r \\ \text{subject to} & \|x_i - x_j\|_2^2 \geq 4r^2, \quad i = 1, \ldots, n-1, \quad j = i, \ldots n \\ & x_i \preceq \mathbf{1}(l - r), \quad i = 1, \ldots, n \\ & x_i \succeq \mathbf{1}(r), \quad i = 1, \ldots, n, \end{array} \tag{7}$$

where x and r are optimization parameters, n and l are problem data, and $\mathbf{1}$ is the vector of ones. Note that maximizing r is the same as minimizing $-r$ and, by monotonicity, maximizing $n\pi r^2$ for $r \geq 0$.

Initialization procedure. For this example we draw x_0 from the uniform distribution $[0, l] \times [0, l]$. Although it is an optimization parameter, r does not occur in any of the g_i so no r_0 is needed.

Small problem instance. In our examples we take $l = 10$, without loss of generality, and take $n = 41$. We use algorithm 3.1 with $\tau_0 = 1$, $\mu = 1.5$, and $\tau_{\max} = 10,000$, which was never reached.

Small problem instance computational details. The subproblems were solved using CVXPY [DCB14] as the interface to the ECOS solver [DCB13] on a 2.20GHZ Intel Xeon processor. The algorithm took between 6 and 20 steps depending on the initialization with an average of 14. steps, and an average solve time for each subproblem of under 2 tenths of a seconds.

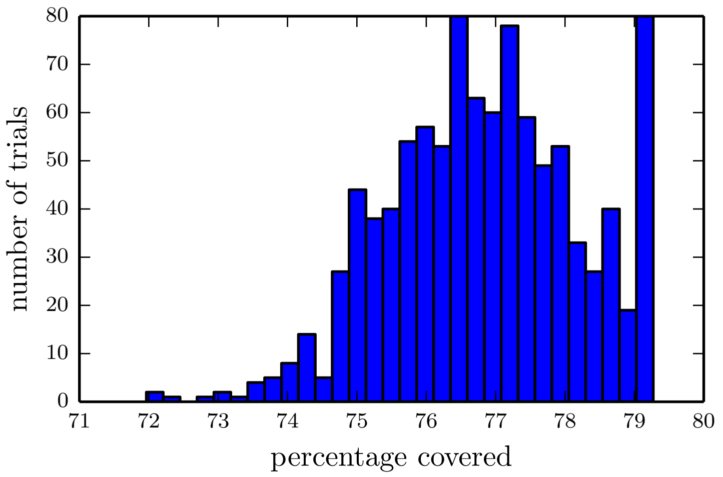

Figure 3: Histogram of the percentage of a square covered for the circle packing problem with 41 circles over 1000 random initializations.

Small problem instance results. Figure 3 shows a histogram of the coverages found for this problem over 1000 initializations. In 14.0 percent of the cases we found a packing within 1 percent of the best known packing for 41 circles of 79.273 percent. The densest packing found by the algorithm of 79.272 percent is shown in figure 4. In 0.3 percent of the cases the algorithm failed due to numerical issues.

Algorithm variation. The number of non-intersection constraints for the circle packing problem grows as n^2, so for large n it may be impossible to impose all of the constraints. We note that for any configuration with $0 < x_i < l$ with x_i distinct for all i, the configuration can be made feasible with sufficiently small r. We therefore enforce the boundary constraints without slack variables at all iterations for numerical stability. We apply the remaining constraints with slack variables using a cutting plane method which includes, at each iteration, the $22n$ constraints with the smallest margin or all currently violated constraints, whichever set is larger. These $22n$ (or more) constraints represent the constraints currently violated or most likely to be violated at the next iteration. This simple method does not have a guarantee of convergence, and more sophisticated cutting plane procedures can be found in the references in §3.3. This method is sufficient for our example.

Large problem instance. We tested the algorithm using $n = 400$ so that approximately 13 percent of all of the constraints were included at each iteration. We set $l = 10$, $\tau_0 = 0.001$, $\mu = 1.05$, and $\tau_{\max} = 10,000$, which was never reached.

Large problem instance computational details. The subproblems were solved using CVXPY [DCB14] as the interface to the ECOS solver [DCB13] on a 2.20GHZ Intel Xeon

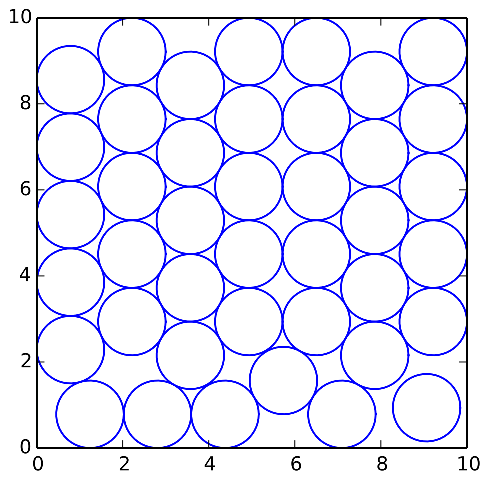

Figure 4: Densest packing found for 41 circles: 79.27.

processor. The algorithm took between 86 and 160 steps depending on the initialization with an average of 125 steps, and an average solve time for each subproblem of under 4 seconds.

Large problem instance results. Figure 6 shows a histogram of the coverages found for this problem over 450 random initializations. In 84.2 percent of cases we were within 3 percent of the best known packing of 86.28 percent coverage. In 28.0 percent of cases we were within 2 percent of the best known packing. The densest packing we found shown in figure 5 is 85.79 percent, within 0.6 percent of the densest known packing. Given that this is a general purpose algorithm, almost always getting within 3 percent is significant. In less than 1 percent of cases, the algorithm failed due to numerical issues.

5.3 Circuit layout

General description. In the circuit layout problem we place components on a chip to minimize the wire length between components. This is a well developed field with an entire industry and numerous books devoted to it. For early work in the field see [HK72] while more recent work can be found in [NC07]. A description of the circuit placement problem we consider can be found in [BV04, §8.8].

Mathematical description. For our components we will consider n square components which may not be rotated. Let x_i, y_i, and l_i for $i = 1, \ldots, n$ be the x position, y position, and side length respectively of component i. The chip is a rectangle with side lengths b_x and b_y. We are given a set of pairs $\mathcal{E}$ representing the connections between the components, and we would like to minimize the ℓ_1 distance (wire length) between the components such that none of the components overlap. In order for two components not to overlap they must be

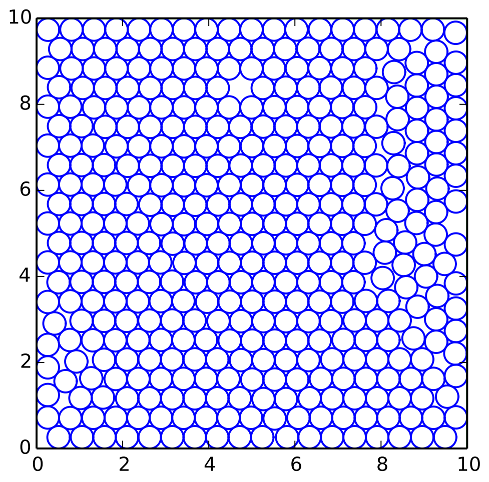

Figure 5: Densest packing found for 400 circles: 85.79.

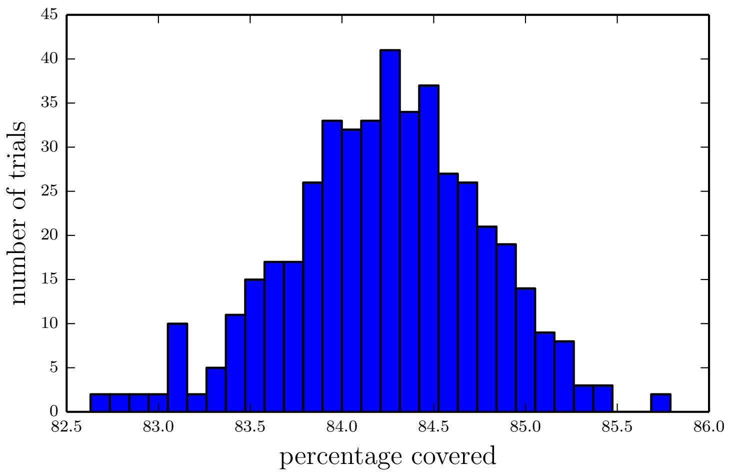

Figure 6: Histogram of the percentage of a square covered for the circle packing problem with 400 circles over 30 random initializations.

either far enough apart in the x direction or the y direction, *e.g.*,

$$|x_i - x_j| \geq \frac{l_i + l_j}{2} \quad \text{or} \quad |y_i - y_j| \geq \frac{l_i + l_j}{2} \quad i = 1, \ldots, n-1, \quad j = i, \ldots, n.$$

We can therefore express the circuit placement problem as

$$\begin{array}{ll} \text{minimize} & \sum_{(i,j)\in\mathcal{E}} |x_i - x_j| + |y_i - y_j| \\ \text{subject to} & \min\left(\frac{l_i+l_j}{2} - |x_i - x_j|, \frac{l_i+l_j}{2} - |y_i - y_j|\right) \leq 0, \quad i = 1, \ldots, n-1, \quad j = i+1, \ldots, n \\ & |x_i| \leq (b_x - l_i)/2, \quad i = 1, \ldots, n \\ & |y_i| \leq (b_y - l_i)/2, \quad i = 1, \ldots, n, \end{array}$$

where x_i and y_i are the optimization parameters and l_i, b_x, b_y, and the connectivity graph are problem data. The objective is convex, and the first constraint is the minimum of concave functions and is therefore concave.

Initialization procedure. We will use two initialization procedures. One good initialization procedure for this algorithm is to use the embedding of the graph Laplacian of the circuit's connections as introduced in [BN03]. In this method the eigenvectors corresponding to the two smallest nonzero eigenvalues of the graph Laplacian are used to initialize the x_i and y_i. We will also initialize x_i and y_i uniformly on $[-b_x/2, b_x/2]$ and $[-b_y/2, b_y/2]$ respectively.

Algorithm variation. We observe that when two components are touching at a corner, the non-overlap constraint is not differentiable, and therefore a subgradient must be chosen. Any linear separator between the vertical and horizontal separator is a valid subgradient. In breaking ties we choose the vertical separator for even values of k and the horizontal separator for odd values of k. By aligning subgradients at each iteration, we allow components to easily slide past each other.

Problem instance. To demonstrate the algorithm we generated an Erdös-Renyi connectivity graph for $n = 15$ components with average degree 6. We took $l_i = 1$, $b_x = b_y = 7$, $\tau_0 = 0.2$, $\mu = 1.1$, and $\tau_{\max} = 10000$.

Computational details. The subproblems were solved using CVX [CR12, GB08] as the interface to the SDPT3 solver [TTT99, TTT03] on a 2.66 GHz Intel Core 2 Duo machine. The algorithm took between 34 and 50 steps depending on the initialization with an average of 41 steps, and an average solve time for each subproblem of 0.29 seconds.

Results. Although this problem was too large for us to solve a mixed integer linear programming representation of it, using the observation that the first four connections to a given component have at least wire length 1, and the next 4 at least wire length 2, we can lower bound the optimal value by 42. Figure 7 shows the best solution we were able to find,

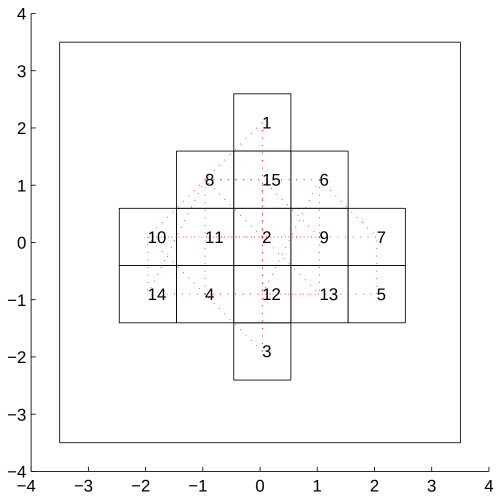

Figure 7: Best circuit layout found for the Erdös-Renyi random graph with average degree 6. Components that are connected are represented by dotted lines.

which has wire length 57. Connected components are signaled by a red dotted lines (note, these are not the paths of the wires). Figure 8 shows a histogram of the wire lengths found over 1000 random initializations . Using the Laplacian initialization finds a layout with wire length 60.

5.4 Multi-matrix principal component analysis

General description. Principal component analysis (PCA) finds the orthogonal directions in data with the greatest variance (and therefore the most significance). Multi-matrix PCA is similar, except that the data is not known exactly, but rather a set (often drawn from a distribution) of possible data is known. Multi-matrix PCA then looks for directions of significance across all of the possible data sets.

Mathematical description. The multi-matrix PCA problem is

$$\begin{array}{ll} \text{maximize} & \min_{i=1,\ldots,p} \mathbf{Tr}\left(X^T A_i X\right) \\ \text{subject to} & X^T X = I, \end{array} \tag{8}$$

where $X \in \mathbf{R}^{n \times m}$ is the optimization variable, $A_i \in \mathbf{S}_+^n$, where $\mathbf{S}_+$ is the set of $n \times n$ positive semidefinite matrices, and I is the identity matrix. The equality constraint is also known as the Steifel manifold [Sti36], and has its own history of optimization techniques; see, *e.g.*, [ETS98, AMS09].

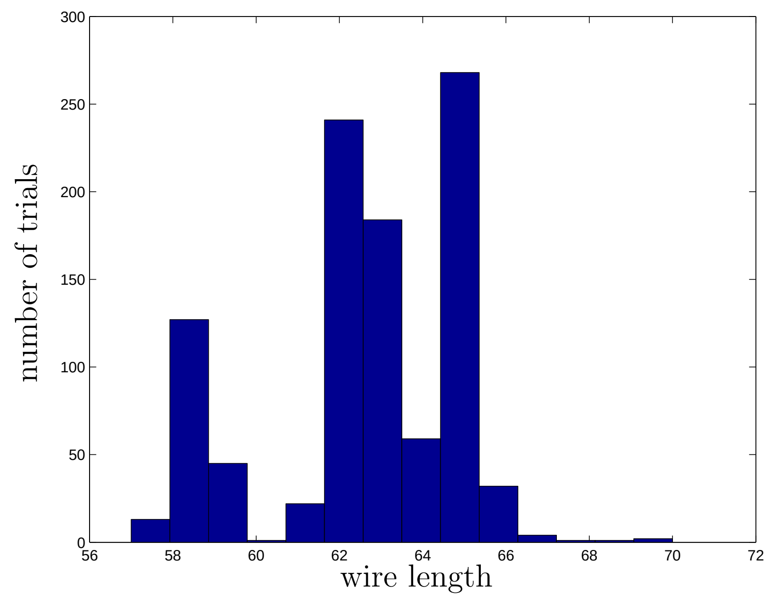

Figure 8: Histogram of wirelengths found over 1000 random initializations.

Problem (8) is equivalent to

$$\begin{array}{ll} \text{maximize} & \min_{i=1,\ldots,p} \mathbf{Tr}\left(X^T\left(A_i - \lambda I\right)X\right) + m\lambda \\ \text{subject to} & X^TX = I, \end{array}$$

where X is the optimization parameter and λ is a scalar. From this we can see that, without loss of generality, we can assume that all of the A_i in (8) are negative definite by choosing λ to be the larger than the largest eigenvalue of any A_i. Therefore we can represent (8) in DC form as

$$\begin{array}{ll} \text{minimize} & -\min_{i=1,\ldots,p} \mathbf{Tr}\left(X^TA_iX\right) \\ \text{subject to} & X^TX - I \preceq 0, \\ & I - X^TX \preceq 0, \end{array}$$

where X is the optimization parameter, the A_i are negative definite, and $\preceq$ is with respect to the positive semidefinite cone. The objective is the negative of a concave function, which is the minimum of concave functions, and is therefore convex, and $X^TX - I$ is convex with respect to the semidefinite cone.

Initialization procedure. We look at two initialization procedures in addition to random initialization. It is well known for the case when $p = 1$ that the principal components can be found by looking at the singular vectors of A corresponding to the m largest singular values. We can therefore calculate an X_i for each of the A_i by solving the PCA problem. We can then set X_0 to be the X_i with the best objective value for (8). Another initialization for X_0 we will look at is to use the solution to PCA using the average of the A_i as X_0.

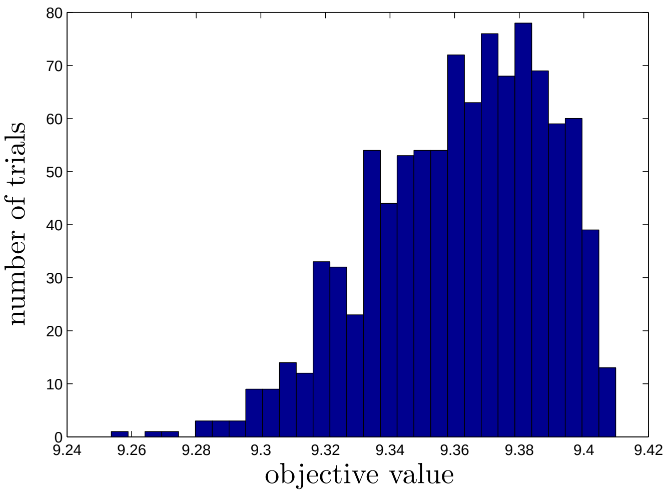

Figure 9: Histogram of the objective value of multi-matrix PCA over 1000 initializations.

Problem instance. For our example we generate a positive definite matrix by creating a diagonal matrix with entries drawn uniformly from the interval $[0, 1]$, and then apply a random orthogonal transformation. We then generate the A_i by varying the entries by up to 50 percent by drawing uniformly from the interval $[-50, 50]$. We then verified that the resulting matrix is positive definite. We used algorithm 4.2 with $m = 10$, $n = 100$, $p = 8$, $\tau_0 = 0.5I$, $\tau_{\mathrm{inc}} = 1.05$, $\tau_{\max} = 10,000$. Observe that the positive semidefinite cone is self dual, and that $0.5I$ is clearly on the interior of the semidefinite cone.

Computational details. The subproblems were solved using CVX [CR12, GB08] as the interface to the SDPT3 solver [TTT99, TTT03] on a 2.66 GHz Intel Core 2 Duo machine. The algorithm took between 62 and 84 steps depending on the initialization with an average of 72 steps, and an average solve time for each subproblem of 35.74 seconds.

Results. Clearly the solution to (8) cannot be larger than the smallest solution for any particular A_i, so we can upper bound the optimal value by 11.10. Using the best X found by solving PCA individually for each of the A_i yields an objective of 7.66 and solving PCA with the average of the A_i yields an objective value of 8.66,. Initializing the algorithm using these values yields 9.33 and 9.41, respectively. The best value found over 1000 random initializations was also 9.41. A histogram of the results can be seen in figure 9.

6 Acknowledgements

This research was made possible by the National Science Foundation Graduate Research Fellowship, grant DGE-1147470 and by the Cleve B. Moler Stanford Graduate Fellowship.

References

[Agi66] N. Agin. Optimum seeking with branch and bound. *Management Science*, 13:176–185, 1966.

[AMS09] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

[BGN00] R. H. Byrd, J. C. Gilbert, and J. Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89(1):149–185, 2000.

[BHÅ14] S. Boyd, M. Hast, and K. J. Åström. MIMO PID tuning via iterated LMI restriction. `https://web.stanford.edu/~boyd/papers/pdf/mimo_pid_tuning.pdf`, 2014.

[BN03] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[BT95] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta Numerica*, 4(1):1–51, 1995.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.

[Byr00] C. Byrne. Block-iterative interior point optimization methods for image reconstruction from limited data. *Inverse Problems*, 16(5):1405, 2000.

[CA96] J. M. Crawford and L. D. Auton. Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81(1):31–57, 1996.

[CR12] Inc. CVX Research. CVX: Matlab software for disciplined convex programming, version 2.0. `http://cvxr.com.cvx`, aug 2012.

[DCB13] A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *European Control Conference*, pages 3071–3076, 2013.

[DCB14] S. Diamond, E. Chu, and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization, version 0.2. `http://cvxpy.org/`, May 2014.

[DL77] J. De Leeuw. Applications of convex analysis to multidimensional scaling. In R. R. Barra, F. Bordeau, G. Romier, and B. Ban Cutsem, editors, *Recent Developments in Statistics*, pages 133–146. North Holland Publishing, Amsterdam, 1977.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[dMVDH99] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1):229–2–37, 1999.

[DPG89] G. Di Pillo and L. Grippo. Exact penalty functions in constrained optimization. *SIAM Journal on Control and Optimization*, 27(6):1333–1360, 1989.

[EM75] J. Elzinga and T. J. Moore. A central cutting plane algorithm for convex programming problems. *Mathematical Programming*, 8:134–145, 1975.

[ETS98] A. Edelman, A. A. Tomás, and T. S. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[FH76] J. E. Falk and K. R. Hoffmann. A successive underestimation method for concave minimization problems. *Mathematics of Operations Research*, 1(3):251–259, 1976.

[FS69] J. E. Falk and R. M. Soland. An algorithm for separable nonconvex programming problems. *Management Science*, 15(9):550–569, 1969.

[GB08] M. Grant and S. Boyd. Graph implementation for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences. Springer-Verlag Limited, 2008. `http://stanford.edu/~boyd/graph_dcp.html`.

[GPM76] U. M. Garcia Palomares and O. L. Mangasarian. Superlinearly convergent quasi-newton algorithms for nonlinearly constrained optimization problems. *Mathematical Programming*, 11(1):1–13, 1976.

[GW12] P. E. Gill and E. Wong. Sequential quadratic programming methods. In *Mixed Integer Nonlinear Programming*, pages 147–224. Springer, 2012.

[Har59] P. Hartman. On functions representable as a difference of convex functions. *Pacific Journal of Math*, 9(3):707–713, 1959.

[Hil75] R. J. Hillestad. Optimization problems subject to a budget constraint with economies of scale. *Operations Research*, 23(6):1091–1098, 1975.

[HJ80a] R. J. Hillestad and S. E. Jacobsen. Linear programs with an additional reverse convex constraint. *Applied Mathematics and Optimization*, 6(1):257–269, 1980.

[HJ80b] R. J. Hillestad and S. E. Jacobsen. Reverse convex programming. *Applied Mathematics and Optimization*, 6(1):63–78, 1980.

[HK72] M. Hanan and J. Kurtzberg. Placement techniques. In M. A. Breuer, editor, *Design Automation of Digital Systems*, volume 1, pages 213–282. Prentice-Hall, 1972.

[HM79] S.-P. Han and O. L. Mangasarian. Exact penalty functions in nonlinear programming. *Mathematical Programming*, 17(1):251–269, 1979.

[Hor86] R. Horst. A general class of branch-and-bound methods in global optimization with some new approaches for concave minimization. *Journal of Optimization Theory and Applications*, 51(2):271–291, 1986.

[HPT95] R. Horst, P. M. Pardalos, and N. V. Thoai. *Introduction to Global Optimization*. Kluwer Academic Publishers, Dordrecht, Netherlands, 1995.

[HPTDV91] R. Horst, T. Q. Phong, N. V. Thoai, and J. De Vries. On solving a DC programming problem by a sequence of linear programs. *Journal of Global Optimization*, 1(2):183–203, 1991.

[HT96] R. Horst and H. Tuy. *Global Optimization*. Springer, New York, New York, third edition, 1996.

[HT99] R. Horst and N. V. Thoai. DC programming: overview. *Journal of Optimization Theory and Applications*, 103(1):1–43, 1999.

[HTB91] R. Horst, N. V. Thoai, and H. P. Benson. Concave minimization via conical partitions and polyhedral outer approximation. *Mathematical Programming*, 50:259–274, 1991.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computation*, pages 85–104. Plenum, 1972.

[Kel60] J. E. Kelly, Jr. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial & Applied Mathematics*, 8(4):703–712, 1960.

[Lan04] K. Lange. *Optimization*. Springer Texts in Statistics. Springer, New York, New York, 2004.

[LHY00] K. Lange, D. R. Hunter, and I. Yang. Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, 9(1):1–20, 2000.

[LR87] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, New York, New York, 1987.

[LS09] G. R. Lanckreit and B. K. Sriperumbudur. On the convergence of the concave-convex procedure. In *Advances in Neural Information Processing Systems*, pages 1759–1767, 2009.

[LW66] E. L. Lawler and D. E. Wood. Branch-and-bound methods: a survey. *Operations Research*, 14:699–719, 1966.

[MB09] A. Mutapcic and S. Boyd. Cutting-set methods for robust convex optimization with pessimizing oracles. *Optimization Methods and Software*, 24(3):381–406, 2009.

[Mey70] R. Meyer. The validity of a family of optimization methods. *SIAM Journal on Control*, 8(1):41–54, 1970.

[MF97] C. D. Maranas and C. A. Floudas. Global optimization in generalized geometric programming. *Computers & Chemical Engineering*, 21(4):351–369, 1997.

[MK07] G. McLachlan and T. Krishnan. *The EM algorithm and extensions*. John Wiley & Sons, 2007.

[ML08] J. B. Mueller and R. Larsson. Collision avoidance maneuver planning with robust optimization. In *International ESA Conference on Guidance, Navigation and Control Systems, Tralee, County Kerry, Ireland*, 2008.

[MOS13] MOSEK ApS. MOSEK version 7.0. http://www.mosek.com, 2013.

[MSL92] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions and SAT problems. In *Proceedings of the tenth National Conference on Artificial Intelligence*, volume 92, pages 459–465, 1992.

[Muu85] L. D. Muu. A convergent algorithm for solving linear programs with an additional reverse convex constraint. *Kybernetika*, 21(6):428–435, 1985.

[NC07] G-J Nam and J. Cong, editors. *Modern Circuit Placement*. Springer, 2007.

[NW06] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2006.

[Pow78] M. J. D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. *Numerical Analysis*, pages 144–157, 1978.

[Rob72] S. M. Robinson. A quadratically-convergent algorithm for general nonlinear programming problems. *Mathematical Programming*, 3(1):145–156, 1972.

[SLA$^+$13] J. Shulman, A. Lee, I. Awwal, H. Bradlow, and P. Abbell. Finding locally optimal collision-free trajectories with sequential convex optimization. *Submitted. Draft at https://sites. google. com/site/rss2013trajopt*, 2013.

[Sol71] R. M. Soland. An algorithm for separable nonconvex programming problems ii: Nonconvex constraints. *Management Science*, 17(11):759–773, 1971.

[Spe13] E. Specht. Packomania. http://www.packomania.com/, October 2013.

[Sti36] E. Stiefel. Richtungsfelder und fernparallelismus in n-dimensionalem mannig faltigkeiten. *Commentarii Mathematici Helvetici*, 8(305–353), 1935–1936.

[Sva87] K. Svanberg. The method of moving asymptotes–a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24(2):359–373, 1987.

[SVH05] A. J. Smola, S. V. N. Vishwanathan, and T. Hofmann. Kernel methods for missing variables. In *Proceedings of 10th international workshop on artificial intelligence and statistics*, 2005.

[TH88] H. Tuy and R. Horst. Convergence and restart in branch-and-bound algorithms for global optimization. Applications to concave minimization and DC optimization problems. *Mathematical Programming*, 41(2):161–183, 1988.

[TT80] N. V. Thoai and H. Tuy. Convergent algorithms for minimizing a concave function. *Mathematics of Operations Research*, 5(4):556–566, 1980.

[TTT99] K. C. Toh, M. J. Todd, and R. H. Tutuncu. SDPT3 — a MATLAB software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1999.

[TTT03] R. H. Tutuncu, K. C. Toh, and M. J. Todd. Solving semidfinite-quadratic-linear programs using SDPT3. *Mathematical Programming*, 95(2):189–217, 2003.

[Tuy83] H. Tuy. On outer approximation methods for solving concave minimization problems. *Acta Mathematica Vietnamica*, 8(2):3–34, 1983.

[Tuy86] H. Tuy. A general deterministic approach to global optimization via D.C. programming. *North-Holland Mathematics Studies*, 129:273–303, 1986.

[Wil63] R. B. Wilson. *A Simplicial Algorithm for Concave Programming*. PhD thesis, Gradaute School of Business Administration, Harvard University, 1963.

[YJ09] C.-N. J Yu and T. Joachims. Learning structural SVMs with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1169–1176, 2009.

[YR03] A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4):915–936, 2003.

[YTI00] S. Yamada, T. Tanino, and M. Inuiguchi. Inner approximation method for a reverse convex programming problem. *Journal of optimization theory and applications*, 107(2):355–389, 2000.

[Zan69] W. I. Zangwill. *Nonlinear Programming: A Unified Approach*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1969.

[Zil01] C. Zillober. Global convergence of a nonlinear programming method using convex optimization. *Numerical Algorithms*, 27(3):265–289, 2001.