Optim Eng (2016) 17:263–287
DOI 10.1007/s11081-015-9294-x

Variations and extension of the convex–concave procedure

Thomas Lipp[1] · **Stephen Boyd**[2]

Received: 20 July 2014 / Revised: 20 February 2015 / Accepted: 10 October 2015 /
Published online: 5 November 2015

Abstract We investigate the convex–concave procedure, 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.

Keywords Convex optimization · Convex concave procedure · Sequential optimization · Difference of convex programming

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. It is worth noting that CCP is very similar to work done on DC algorithms (DCA), which we will discuss in detail later. For ease

Electronic supplementary material The online version of this article (doi:10.1007/s11081-015-9294-x) contains supplementary material, which is available to authorized users.

✉ Thomas Lipp
tlipp@stanford.edu

Stephen Boyd
boyd@stanford.edu

[1] Stanford University, Packard Building, Room 243, 350 Serra Mall, Stanford, CA 94305, USA

[2] Stanford University, Packard Building, Room 254, 350 Serra Mall, Stanford, CA 94305, USA

of presentation, we believe greater clarity comes from taking CCP as a starting point rather than DCA, although similar points could be reached by progressing through the DCA literature, and there is much overlap. We will reference results in the DCA literature when appropriate.

1.1 Difference of convex programming

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

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

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 $i = 0, \ldots, m$ 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 (Hartman 1959). 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)

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

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

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

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 (Karp 1972). We will examine one instance from this class, 3-satisfiability, in Sect. 5.

The global solution to (1) can be found through general branch and bound methods (Agin 1966; Lawler and Wood 1966). 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 (Nocedal and Wright 2006).

1.2 Convex–concave procedure

We now present the basic convex–concave procedure, also known as the concave-convex procedure (Yuille and Rangarajan 2003). 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.*,

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

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

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

is convex, since the objective and constraint functions are convex, and can therefore be solved efficiently (assuming the functions f_i can be handled tractably).

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 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; taking a larger step may lead to faster convergence.

Algorithm 1.2 *Line search for CCP.*

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

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

$$f_i(x_k) - \hat{g}_i(x_k; x_k) = f_i(x_k) - g_i(x_k) \leq 0,$$

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. The above analysis holds in the nondifferentiable case when the gradient is replaced by a subgradient, that is any $\delta g_i(x_k)$ such that for all x,

$$g_i(x_k) + \delta g_i(x_k)(x - x_k) \leq g_i(x).$$

A proof showing convergence to critical points of the original problem in the differentiable case can be found in (Lanckreit and Sriperumbudur 2009).

Although the objective value converges in all cases, 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 (Byrd et al. 2000). 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 methods and algorithms for solving more general convex programs have improved it has become beneficial to take advantage of greater information.

The CCP algorithm above can be derived from DCA, but we prefer this formulation as it is a purely primal description of the problem, *i.e.*, it makes no reference to a dual problem or conjugate functions.

1.5 Outline

In Sect. 2 we examine previous work in solving DC programs and the history of iterative convexification procedures. In Sect. 3 we will introduce our first extension to CCP in which constraints are loosened. In Sect. 4 we present our second extension of CCP, giving a vector version of the algorithm, which is particularly relevant for matrix constraints. In Sect. 5 we present several examples using these methods including 3-satisfiability, circle packing, placement, 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) (Tuy 1986; Tuy and Horst 1988; Horst et al. 1991a). Good overviews of the work that has been done in solving DC programs globally can be found in (Horst et al. 1995; Horst and Thoai 1999), 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 (Maranas and Floudas 1997). Branch and bound methods were

originally popularized for combinatorial problems (Agin 1966; Lawler and Wood 1966) but soon made the transition to general nonconvex optimization (Falk and Soland 1969; Horst 1986; Soland 1971). 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 (Meyer 1970), or concave minimization problems. These problems are often approached with simplicial algorithms as in (Hillestad 1975; Hillestad and Jacobsen 1980a) or cutting plane methods as in (Hillestad and Jacobsen 1980b; Thoai and Tuy 1980; Muu 1985). Cutting plane methods, an early optimization technique as seen in (Zangwill 1969), 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 (Falk and Hoffmann 1976; Tuy 1983; Horst et al. 1991b) and the less common inner approximation as in (Yamada et al. 2000). A more in depth discussion of these problem classes and approaches can be found in (Horst and Tuy 1996).

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 (Wilson 1963) with convergence properties shown in (Robinson 1972). 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 (Boggs and Tolle 1995; Nocedal and Wright 2006; Gill and Wong 2012) 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 (Dempster et al. 1977) and although MM algorithms are just as old, as seen in (De Leeuw 1977), the term majorization minimization was not coined until Hunter and Lange's rejoinder to (Lange et al. 2000). 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 (Little and Rubin 1987; Lange 2004; McLachlan and Krishnan 2007). MM approaches have been employed frequently in signal processing applications as seen in (Stoica and Babu 2012), with approaches that have started to address the DC programming problem as in (Lange et al. 2000; Naghsh et al. 2013).

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 (Svanberg 1987) and later expanded in (Zillober 2001) this method similarly involves sequential convexification, although the parameters are rescaled to drive solutions away from variable limits. Also in structural engineering there is sequential parametric convex approximation (Beck et al. 2010). In vehicle avoidance and trajectory problems, many have independently developed their own sequential convexification procedures which are effectively CCP procedures. A few examples include (Mueller and Larsson 2008; Shulman et al. 2013).

The work that most closely relates to the work presented here is that on difference of convex algorithms (DCA), presented in (Pham Dinh and Souad 1986) as methods of subgradients, for solving optimization problems where the objective is a DC function. In fact, CCP can be viewed as a version of DCA which instead of explicitly stating the linearization as CCP does, finds it by solving a dual problem. Thus DCA consists of solving an alternating sequence of primal and dual problems. This work was then expanded to include optimization problems with DC constraints in (Pham Dinh and Souad 1988) before settling on the term DCA in (Le Thi et al. 1996; Pham Dinh and Le Thi 1997). DCA has been applied to a variety of applications (Le Thi and Pham Dinh 1997; Pham Dinh and Le Thi 1998; Le Thi and Pham Dinh 2008; Le Thi et al. 2009a, b), including large scale approaches (Le Thi and Pham Dinh 2003; Le Thi 2003; Pham Dinh et al. 2008), and combination with branch and bound methods for global optimization (Le Thi et al. 1998, 2002; Pham Dinh et al. 2010; Le et al. 2010). More recent work has addressed the problem of infeasible constraints which we address in this paper. Indeed the DCA2 algorithm presented in (Le Thi et al. 2014; Pham Dinh and Le Thi 2014) does away with DCA's alternating primal and dual steps to use a simple linearization that is nearly identical to our algorithm 3.1, although we arrive at it from a different direction. There have also been approaches that adapt DCA for conic inequalities (Niu and Pham Dinh 2014), but none seem to provide a general framework as given in Sect. 4. Still more information on DCA can be found at (Le Thi 2015) and the references therein.

The convex–concave procedure was first introduced geometrically in (Yuille and Rangarajan 2003) although without inequality constraints. Our approach and analysis more closely follows that in (Smola et al. 2005) 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 (Byrne 2000), support vector machines (SVM) with additional structure as in (Yu and Joachims 2009), design of feedback gains (Fardad and Jovanović 2014), and even for tuning multi-input multi-output proportional-integral-derivative control as in (Boyd et al. 2015) 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 (Garcia Palomares and Mangasarian 1976; Powell 1978). 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 (Boyd and Vandenberghe 2004), Sect. 6.3.2. Therefore, if we are unable to satisfy all constraints, the set of violated constraints should be small. As mentioned earlier, with the exception of the penalty update, algorithm 3.1 is identical to the algorithm DCA2 mentioned in (Le Thi et al. 2014; Pham Dinh and Le Thi 2014). However, because we arrive at the algorithm through different motivations, we maintain the different name.

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
   $$\begin{array}{ll} \text{minimize} & f_0(x) - \hat{g}_0(x; x_k) + \tau_k \sum_{i=1}^m s_i \\ \text{subject to} & f_i(x) - \hat{g}_i(x; x_k) \leq s_i, \quad i = 1, \ldots, m \\ & s_i \geq 0, \quad i = 1, \ldots, m. \end{array}$$
3. *Update* τ. $\tau_{k+1} := \min(\mu \tau_k, \tau_{\max})$.
4. *Update iteration.* $k := k + 1$.

until stopping criterion is satisfied.

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

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

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

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

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 $\left(e.g., \sum_{i=1}^{m} s_i = 0\right)$ (Han and Mangasarian 1979; Di Pillo and Grippo 1989). 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. For work on penalty functions from the DCA point of view, see (Le Thi et al. 2012). 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 Sect. 5.3.

3.2 Enforcing constraints

There are many variations on how constraints are handled in the algorithm. 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. Our experience in numerical examples suggests that this last variation is ill advised, since it seems to constrain the algorithm prematurely to a region without feasible points, or with higher objective value.

3.3 Cutting plane techniques

Often DC programs may have large numbers of constraints, as we will see in the circle packing problem of Sect. 5.2. However, most of these constraints are inactive, *i.e.*, we have $f_i(x) - g_i(x) \ll 0$. In these instances, cutting plane or column

generation techniques can be used (Elzinga and Moore 1975; du Merle et al. 1999; Mutapcic and Boyd 2009). 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 Sect. 2 for solving concave minimization problems (Kelly 1960). 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 Sect. 2, new constraints need to be generated. An implementation for this latter method for DC problems can be seen in (Ndiaye et al. 2012).

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, $f_i : \mathbf{R}^n \to \mathbf{R}^p$ and $g_i : \mathbf{R}^n \to \mathbf{R}^p$ for $i = 1, \ldots, m$ are K-convex and differentiable, and 0 is the length p vector of zeros. 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 $y - x \in \mathbf{int}\, K$. Note that although this formulation does not limit generality as, if K_1 and K_2 are proper cones, then so is $K = K_1 \times K_2$, things change very little if each inequality has a different cone K_i. Similar to scalar convexity, f is K-convex if for all x and y we have,

$$f(x) \succeq_K f(y) + Df(y)(x - y), \tag{6}$$

where $Df(y)$ is the derivative or Jacobean matrix of f evaluated at y. For more background on generalized inequalities and proper cones see (Boyd and Vandenberghe 2004), Sect. 2.4.1. For more on convexity with respect to generalized inequalities see (Boyd and Vandenberghe 2004), Sect. 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 if g_i is not differentiable, any matrix can be substituted for $Dg_i(x_k)$ which satisfies the property in (6). 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 example, we have

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

For more on convex optimization problems with generalized inequalities see (Boyd and Vandenberghe 2004), Sects. 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 $\|\mu t_k\|_2 \leq \tau_{\max}$
   $t_{k+1} := \mu t_k$,
   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 $i \geq 0$. Note that because $s_i \succeq_K 0$, $t_k^T s_i \geq 0$ at all iterations, because s_i is in a proper cone, and t_k is in its dual. As in algorithm 3.1, our objective increases the cost of violating the inequality at each iterate, driving $t_k^T s_i$ towards zero. Observe that if $t_k^T s_i = 0$, then $t_k^T s_i \leq 0$ 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. For more information on dual cones and generalized inequalities see (Boyd and Vandenberghe 2004), Sect. 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 specific applications. It is not our intention to compete with these problem-specific algorithms, but rather to show that a general approach using CCP, with no or minimal tuning, performs remarkably well in a variety of settings. In each case simple modifications could be made to to improve CCP's performance, but these examples serve to illustrate an implementation of CCP and highlight several features. Example code for these problems is available at (Lipp and Boyd 2014).

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 of variables and, possibly, negations of variables.

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

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

There is no objective since we are only looking for a feasible point; c is the zero vector. There are many different DC representations for Boolean variables, besides the one given in problem (2), and more can be read about these variations in (Pham Dinh and Le Thi 2014).

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

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{n} |x_i - 0.5| \\ \text{subject to} & 0 \preceq x \preceq 1, \\ & Ax \preceq b, \end{array} \tag{7}$$

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 (Mitchell et al. 1992) there is a threshold around 4.25 expressions per variable when problems transition from being feasible with high probability to being infeasible with high probability (Crawford and Auton 1996). 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 (MOSEK ApS 2013). 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.

Computational details The subproblems were solved using CVX (CVX Research 2012; Grant and Boyd 2008) as the interface to the SDPT3 solver (Toh et al. 1999; Tutuncu et al. 2003) 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 (7). Using random initializations, satisfying assignments could be found consistently for up to 3.5 constraints per variable at which point success started to decrease.

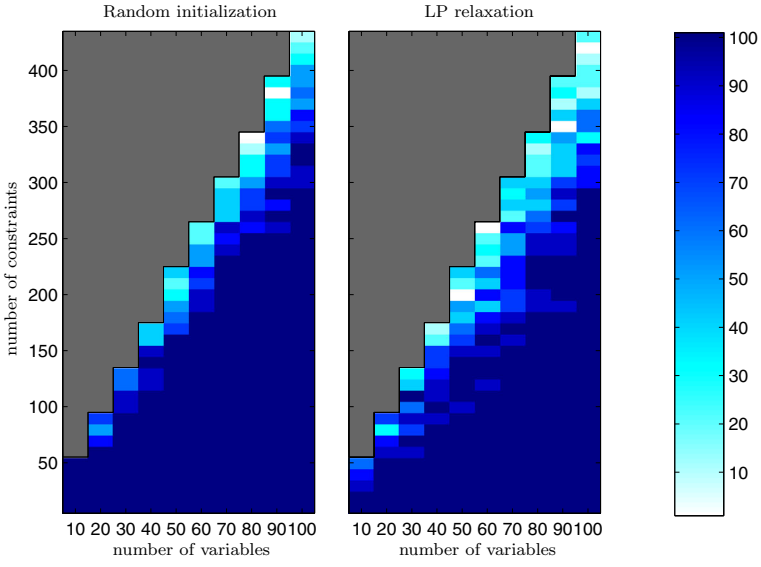

Fig. 1 Percentage of runs for which a satisfying assignment to random 3-SAT problems were found for problems of varying sizes. Problems in the *gray region* are with high probability infeasible

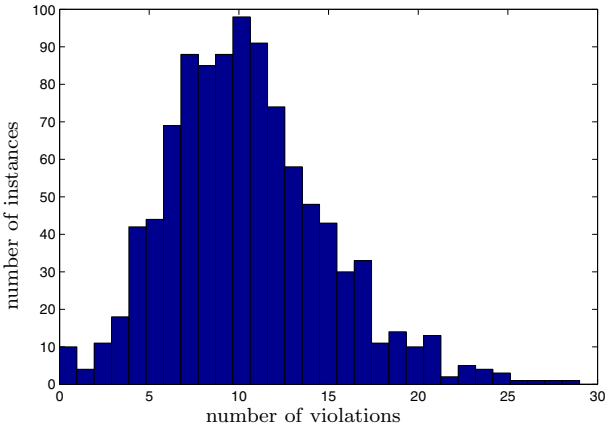

Fig. 2 Histogram of the number of unsatisfied expressions for a hard 3-SAT problem instance with 100 variables and 430 expressions

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; still, we see that even when CCP fails (to find a feasible point) it typically finds points satisfying almost all of the expressions. 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 is to find 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 (Specht 2013).

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 the center of circle i and r be radius of the circles. The problem is

$$\begin{array}{lll} \text{maximize} & r & \\ \text{subject to} & \|x_i - x_j\|_2^2 \geq 4r^2, & i = 1, \ldots, n-1, \quad j = i, \ldots n \\ & x_i \preceq \mathbf{1}(l - r), & i = 1, \ldots, n x_i \succeq \mathbf{1}(r), \quad i = 1, \ldots, n, \end{array} \tag{8}$$

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 $r \geq 0$. The first inequality constraint can be represented in DC form as

$$4r^2 - \|x_i - x_j\|_2^2 \leq 0.$$

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 (Diamond et al. 2014; Diamond and Boyd 2015) as the interface to the ECOS solver (Domahidi et al. 2013) on a 2.20GHZ Intel Xeon processor. We used a direct application of algorithm 3.1. 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 0.2 s.

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

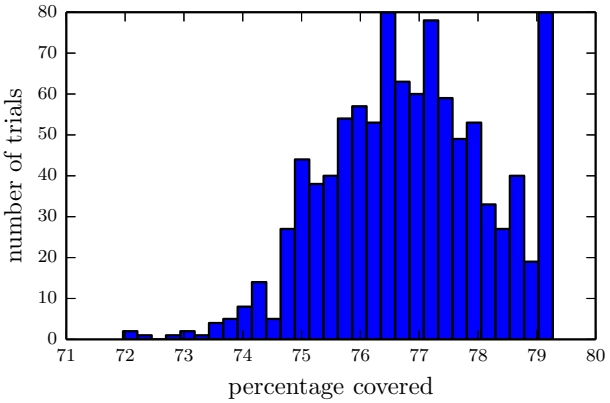

Fig. 3 Histogram of the percentage of a square covered for the circle packing problem with 41 circles over 1000 random initializations

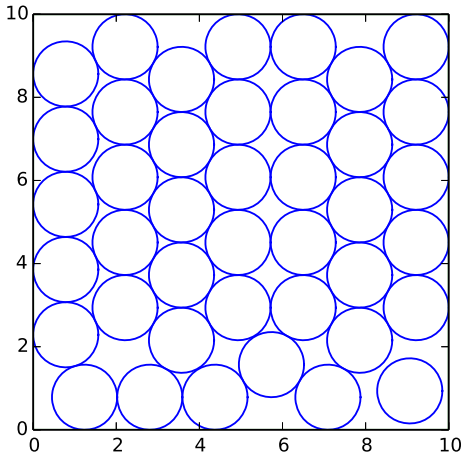

Fig. 4 Densest packing found for 41 circles: 79.27

convergence, and more sophisticated cutting plane procedures can be found in the references in Sect. 3.3. This method is sufficient for our example.

Large problem instance We tested the algorithm using $n = 400$ so that approximately 13 % 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 (Diamond et al. 2014; Diamond and Boyd 2015) as the interface to the ECOS solver (Domahidi et al. 2013) on a 2.20GHZ Intel Xeon 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 % of cases we were within 3 % of the best known packing of 86.28 % coverage. In 28.0 % of cases we were within 2 % of the best known packing. The densest packing we found, shown in Fig. 5, is 85.79 %, within 0.6 % of the densest known packing. Given that this is a general purpose algorithm, almost always getting within 3 % is significant. In less than 1 % of cases, the algorithm failed due to numerical issues.

5.3 Placement

General description In the placement problem (also called floor-planning or layout) we arrange non-overlapping components to minimize an objective such as the total distance between specified components. This is a well developed field, used in several applications areas; for early work in the field see (Hanan and Kurtzberg 1972). A description of the placement problem we consider can be found in (Boyd and Vandenberghe 2004), Sect. 8.8. This problem can also be thought of as a greatly simplified circuit problem, where the distance between components can stand in for wire length. (Real circuit placement problems are much more complex;

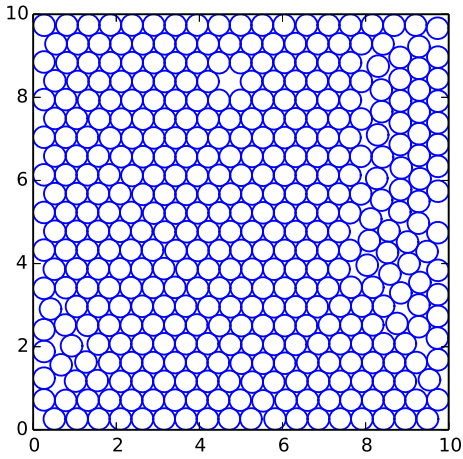

Fig. 5 Densest packing found for 400 circles: 85.79

see (Nam and Cong 2007)) For more on floor planning problems we direct the reader to (Drira et al. 2007; Singh and Sharma 2006) and references therein. There has also been some work on layout and packing type problems in the DC literature as in (Ndiaye et al. 2012).

Mathematical description For our components we will consider n square components. 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 placement area is a rectangle with side lengths b_x and b_y. We are given a set of pairs $\mathcal{E}$ representing the components we wish to be close, and we would like to minimize the ℓ_1 distance between the components such that none of the components overlap. In order for two components not to overlap they must be 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 placement problem as

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

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 (for these constraint $f_i(x) = 0$).

Initialization procedure We will use two initialization procedures. One good initialization procedure for this algorithm is to use the embedding of the graph

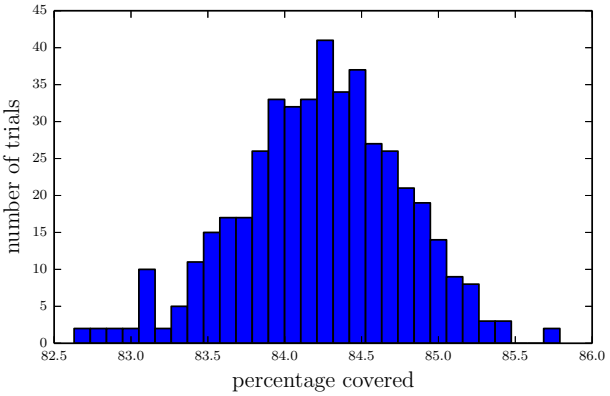

Fig. 6 Histogram of the percentage of a square covered for the circle packing problem with 400 circles over 450 random initializations

Laplacian of the connections as introduced in (Belkin and Niyogi 2003). 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 solved the problem using algorithm 3.1 with a particular choice of subgradients when necessary. 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 $\tau_{\max} = 10{,}000$.

Computational details The subproblems were solved using CVX (CVX Research 2012; Grant and Boyd 2008) as the interface to the SDPT3 solver (Toh et al. 1999; Tutuncu et al. 2003) 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 s.

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 distance 1, and the next 4 at least distance 2, we can lower bound the optimal value by 42. Figure 7 shows the best solution we were able to find, which has wire length 57. Connected components are signaled by a red dotted lines (note, these are not the distances measured). Figure 8 shows a histogram of the distances between components found over 1000 random initializations. Using the Laplacian initialization, CCP finds a layout with distance 60.

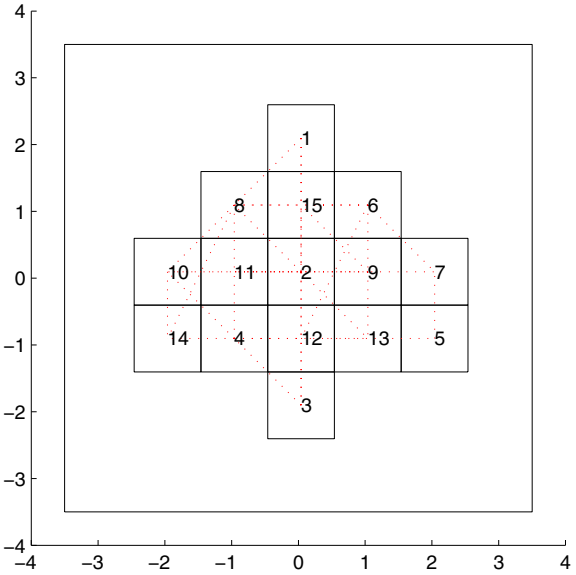

Fig. 7 Best placement found for the Erdös-Renyi random graph with average degree 6. Components that are connected are represented by dotted lines

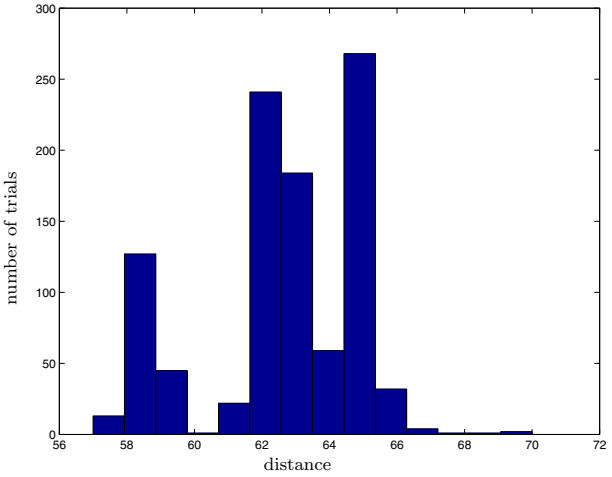

Fig. 8 Histogram of distances found over 1000 random initializations

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

$$\begin{array}{ll}\text{maximize} & \min_{i=1,\ldots,p} \mathbf{Tr}(X^T A_i X) \\ \text{subject to} & X^T X = I,\end{array} \tag{9}$$

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 Stiefel manifold (Stiefel 1935–1936), and has its own history of optimization techniques; see, *e.g.*, (Edelman et al. 1998; Absil et al. 2009).

Problem (9) is equivalent to

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

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 (9) are negative definite by choosing λ to be larger than the largest eigenvalue of any A_i. Therefore we can represent (9) in DC form as

$$\begin{array}{lll}\text{minimize} & & -\min_{i=1,\ldots,p} \mathbf{Tr}(X^T A_i X) \\ \text{subject to} & & X^T X - I \preceq 0, \\ & & I - X^T X \preceq 0,\end{array}$$

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 $X^T X - I$ 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

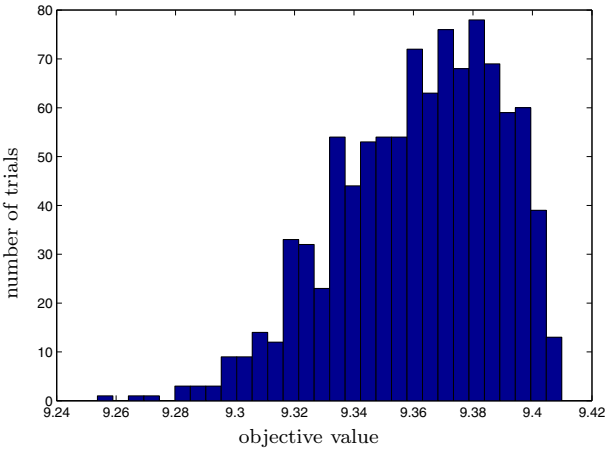

Fig. 9 Histogram of the objective value of multi-matrix PCA over 1000 initializations

solving the PCA problem. We can then set X_0 to be the X_i with the best objective value for (9). 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.

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 % by drawing uniformly from the interval $[-50, 50]$. We then verified that the resulting matrix is positive definite. We used algorithm 4.2 with $m = 10$, $n = 100$, $p = 8$, $\tau_0 = 0.5I$, $\tau_{\text{inc}} = 1.05$, $\tau_{\max} = 10{,}000$. Observe that the positive semidefinite cone is self dual, and that $0.5\,I$ is clearly in the interior of the semidefinite cone.

Computational details The subproblems were solved using CVX (CVX Research 2012; Grant and Boyd 2008) as the interface to the SDPT3 solver (Toh et al. 1999; Tutuncu et al. 2003) 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 s.

Results Clearly the solution to (9) 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 Fig. 9.

6 Conclusion

In this paper we have presented simple heuristics for approaching DC programs, extending the traditional CCP. We have shown through a number of examples that these heuristics work remarkably well, even on hard problems. It is not our intention that these heuristic should compete with state of the art algorithms designed to address the specific examples or even that these are the best CCP approaches for these problems. Rather, we have shown that even with simplistic application of these algorithms, useful results can be achieved, and that these algorithms should certainly be one of the tools in an optimization arsenal.

Acknowledgments We would like to thank our many reviewers for their comments which improved this paper and in particular for highlighting much of the recent work in DCA. 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

Absil PA, Mahony R, Sepulchre R (2009) Optimization algorithms on matrix manifolds. Princeton University Press, Princeton

Agin N (1966) Optimum seeking with branch and bound. Manag Sci 13:176–185

Beck A, Ben-Tal A, Tetrushvili L (2010) A sequential parametric convex approximation method with applications to nonconvex truss topology design problems. J Glob Optim 47(1):29–51
Belkin M, Niyogi P (2003) Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput 15(6):1373–1396
Boggs PT, Tolle JW (1995) Sequential quadratic programming. Acta Numer 4(1):1–51
Boyd S, Vandenberghe L (2004) Convex Optim. Cambridge University Press, Cambridge
Boyd S, Hast M, Åström KJ (2015) MIMO PID tuning via iterated LMI restriction. Int J Robust Nonlinear Control To appear
Byrd RH, Gilbert JC, Nocedal J (2000) A trust region method based on interior point techniques for nonlinear programming. Math Program 89(1):149–185
Byrne C (2000) Block-iterative interior point optimization methods for image reconstruction from limited data. Inverse Probl 16(5):1405
Crawford JM, Auton LD (1996) Experimental results on the crossover point in random 3-SAT. Artif Intell 81(1):31–57
CVX Research I (2012) CVX: MATLAB software for disciplined convex programming, version 2.0. http://cvxr.com.cvx
De Leeuw J (1977) Applications of convex analysis to multidimensional scaling. In: Barra JR, Brodeau F, Romier G, Van Cutsem B (eds) Recent developments in statistics. North Holland Publishing, Amsterdam, pp 133–146
Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc Ser B (Methodol) 39:1–38
Di Pillo G, Grippo L (1989) Exact penalty functions in constrained optimization. SIAM J Control Optim 27(6):1333–1360
Diamond S, Boyd S (2015) CVXPY: a python-embedded modeling language for convex optimization. J Mach Learn Res Mach Learn Open Sour Softw To appear
Diamond S, Chu E, Boyd S (2014) CVXPY: a python-embedded modeling language for convex optimization, version 0.2. http://cvxpy.org/
Domahidi A, Chu E, Boyd S (2013) ECOS: an SOCP solver for embedded systems. In: European control conference, pp 3071–3076
Drira A, Pierreval H, Hajri-Gabouh S (2007) Facility layout problems: a survey. Annu Rev Control 31(2):255–267
du Merle O, Villeneuve D, Desrosiers J, Hansen P (1999) Stabilized column generation. Discret Math 194(1):229–237
Edelman A, Tomás AA, Smith TS (1998) The geometry of algorithms with orthogonality constraints. SIAM J Matrix Anal Appl 20(2):303–353
Elzinga J, Moore TJ (1975) A central cutting plane algorithm for convex programming problems. Math Program 8:134–145
Falk JE, Hoffmann KR (1976) A successive underestimation method for concave minimization problems. Math Oper Res 1(3):251–259
Falk JE, Soland RM (1969) An algorithm for separable nonconvex programming problems. Manag Sci 15(9):550–569
Fardad M, Jovanović MR (2014) On the design of optimal structured and sparse feedback gains via sequential convex programming. In: American control conference, pp 2426–2431
Garcia Palomares UM, Mangasarian OL (1976) Superlinearly convergent quasi-newton algorithms for nonlinearly constrained optimization problems. Math Program 11(1):1–13
Gill PE, Wong E (2012) Sequential quadratic programming methods. In: Mixed integer nonlinear programming, Springer, pp 147–224
Grant M, Boyd S (2008) Graph implementation for nonsmooth convex programs. In: Blondel V, Boyd S, Kimura H (eds) Recent Advances in learning and control, lecture notes in control and information sciences, Springer. http://stanford.edu/ boyd/graph_dcp.html
Han SP, Mangasarian OL (1979) Exact penalty functions in nonlinear programming. Math Program 17(1):251–269
Hanan M, Kurtzberg J (1972) Placement techniques. In: Breuer MA (ed) Design automation of digital systems, vol 1. Prentice-Hall, Upper Saddle River, pp 213–282
Hartman P (1959) On functions representable as a difference of convex functions. Pac J Math 9(3):707–713
Hillestad RJ (1975) Optimization problems subject to a budget constraint with economies of scale. Oper Res 23(6):1091–1098

Hillestad RJ, Jacobsen SE (1980a) Linear programs with an additional reverse convex constraint. Appl Math Optim 6(1):257–269
Hillestad RJ, Jacobsen SE (1980b) Reverse convex programming. Appl Math Optim 6(1):63–78
Horst R (1986) A general class of branch-and-bound methods in global optimization with some new approaches for concave minimization. J Optim Theory Appl 51(2):271–291
Horst R, Thoai NV (1999) DC programming: overview. J Optim Theory Appl 103(1):1–43
Horst R, Tuy H (1996) Global Optimization, 3rd edn. Springer, New York
Horst R, Phong TQ, Thoai NV, De Vries J (1991a) On solving a DC programming problem by a sequence of linear programs. J Glob Optim 1(2):183–203
Horst R, Thoai NV, Benson HP (1991b) Concave minimization via conical partitions and polyhedral outer approximation. Math Program 50:259–274
Horst R, Pardalos PM, Thoai NV (1995) Introduction to global optimization. Kluwer Academic Publishers, Dordrecht
Karp RM (1972) Reducibility among combinatorial problems. In: Thatcher JW, Miller RE (eds) Complexity of computer computation. Plenum, Berlin, pp 85–104
Kelly JE Jr (1960) The cutting-plane method for solving convex programs. J Soc Ind Appl Math 8(4):703–712
Lanckreit GR, Sriperumbudur BK (2009) On the convergence of the concave-convex procedure. Adv Neural Inf Process Syst, 1759–1767
Lange K (2004) Optimization. Springer texts in statistics. Springer, New York
Lange K, Hunter DR, Yang I (2000) Optimization transfer using surrogate objective functions. J Comput Graph Stat 9(1):1–20
Lawler EL, Wood DE (1966) Branch-and-bound methods: a survey. Oper Res 14:699–719
Le HM, Le Thi HA, Pham Dinh T, Bouvry P (2010) A combined DCA: GA for constructing highly nonlinear balanced Boolean functions in cryptography. J Glob Optim 47(4):597–613
Le Thi HA (2003) Solving large scale molecular distance geometry problems by a smoothing technique via the Gaussian transform and DC programming. J Glob Optim 27(4):375–397
Le Thi HA (2015) DC programming and DCA: local and global approaches—theory, algorithms and applications. http://lita.sciences.univ-metz.fr/lethi/DCA.html
Le Thi HA, Pham Dinh T (1997) Solving a class of linearly constrainted indefinite quadratic problems by DC algorithms. J Glob Optim 11(3):253–285
Le Thi HA, Pham Dinh T (2003) Large-scale molecular optimization for distance matrices by a DC optimization approach. SIAM J Optim 14(1):77–114
Le Thi HA, Pham Dinh T (2008) A continuous approach for the concave cost supply problem via DC programming and DCA. Discret Appl Math 156(3):325–338
Le Thi HA, Pham Dinh T, Muu LD (1996) Numerical solution for optimization over the efficient set by d.c. optimization algorithms. Oper Res Lett 19:117–128
Le Thi HA, Pham Dinh T, Muu LD (1998) A combined D.C. optimization-ellipsoidal branch-and-bound algorithm for solving nonconvex quadratic programming problems. J Comb Optim 2(1):9–28
Le Thi HA, Pham Dinh T, Thoai NV (2002) Combination between global and local methods for solving an optimization problem over the efficient set. Eur J Oper Res 142(2):258–270
Le Thi HA, Moeini M, Pham Dinh T (2009a) DC programming approach for portfolio optimization under step increasing transaction costs. Optimization 58(3):267–289
Le Thi HA, Pham Dinh T, Nguyen CN, Thoai NV (2009b) DC programming techniques for solving a class of nonlinear bilevel programs. J Glob Optim 44(3):313–337
Le Thi HA, Pham Dinh T, Ngai HV (2012) Exact penalty and error bounds in DC programming. J Glob Optim 52(3):509–535
Le Thi HA, Huynh VN, Pham Dinh T (2014) DC programming and DCA for general DC programs. Adv Comput Methods Knowl Eng, 15–35
Lipp T, Boyd S (2014) MATLAB and python examples of convex-concave procedure. http://www.stanford.edu/ boyd/software/cvx_ccv_examples/
Little RJA, Rubin DB (1987) Statistical analysis with missing data. Wiley, New York
Maranas CD, Floudas CA (1997) Global optimization in generalized geometric programming. Comput Chem Eng 21(4):351–369
McLachlan G, Krishnan T (2007) The EM algorithm and extensions. Wiley, Hoboken
Meyer R (1970) The validity of a family of optimization methods. SIAM J Control 8(1):41–54
Mitchell D, Selman B, Levesque H (1992) Hard and easy distributions and SAT problems. Proc Tenth Natl Conf Artif Intell 92:459–465

MOSEK ApS (2013) MOSEK version 7.0. http://www.mosek.com
Mueller JB, Larsson R (2008) Collision avoidance maneuver planning with robust optimization. In: International ESA conference on guidance, navigation and control systems, Tralee
Mutapcic A, Boyd S (2009) Cutting-set methods for robust convex optimization with pessimizing oracles. Optim Methods Softw 24(3):381–406
Muu LD (1985) A convergent algorithm for solving linear programs with an additional reverse convex constraint. Kybernetika 21(6):428–435
Naghsh MM, Modarres-Hashemi M, ShahbazPanahi S, Soltanalian M, Stoica P (2013) Unified optimization framework for multi-static radar code design using information-theoretic criteria. IEEE Trans Signal Process 61(21):5401–5416
Nam GJ, Cong J (eds) (2007) Modern circuit placement. Springer, New York
Ndiaye BM, Pham Dinh T, Le Thi HA (2012) DC programming and DCA for large-scale two-dimensional packing problems. Springer, New York
Niu YS, Pham Dinh T (2014) DC programming approaches for BMI and QMI feasibility problems. Adv Comput Methods Knowl Eng 6:37–63
Nocedal J, Wright SJ (2006) Numerical optimization. Springer, New York
Pham Dinh T, Le Thi HA (1997) Convex analysis approach to DC programming: theory, algorithms, and applications. Acta Math Vietnam 22(1):289–355
Pham Dinh T, Le Thi HA (1998) A DC optimization algorithm for solving the trust-region subproblem. SIAM J Optim 8(2):476–505
Pham Dinh T, Le Thi HA (2014) Recent advances in DC programming and DCA. Trans Comput Intell XIII 13:1–37
Pham Dinh T, Souad EB (1986) Algorithms for solving a class of nonconvex optimization problems. Methods of subgradients. In: Hiriart-Urruty JB (ed) FERMAT Days 85: mathematics for optimization. Elsevier Scince Publishers B. V., Amsterdam, pp 249–271
Pham Dinh T, Souad EB (1988) Duality in D.C. (difference of convex functions) optimization. subgradient methods. In: International Series of Numerical Mathematics, vol 84, Birkhäuser Basel
Pham Dinh T, Le Thi HA, Akoa F (2008) Combining DCA (DC algorithms) and interior point techniques for large-scale nonconvex quadratic programming. Optim Methods Softw 23(4):609–629
Pham Dinh T, Nguyen CN, Le Thi HA (2010) An efficient combined DCA and B&B using DC/SDP relaxation for globally solving binary quadratic programs. J Glob Optim 48(4):595–632
Powell MJD (1978) A fast algorithm for nonlinearly constrained optimization calculations. Springer, New York, pp 144–157
Robinson SM (1972) A quadratically-convergent algorithm for general nonlinear programming problems. Math Program 3(1):145–156
Shulman J, Ho J, Lee A, Awwal I, Bradlow H, Abbeel P (2013) Finding locally optimal, collision-free trajectories with sequential convex optimization. In: Robotics: science and systems, vol 9. pp 1–10
Singh SP, Sharma RRK (2006) A review of different approaches to the facility layout problems. Int J Adv Manuf Technol 30(5):425–433
Smola AJ, Vishwanathan SVN, Hofmann T (2005) Kernel methods for missing variables. In: Proceedings of 10th international workshop on artificial intelligence and statistics
Soland RM (1971) An algorithm for separable nonconvex programming problems II: nonconvex constraints. Manag Sci 17(11):759–773
Specht E (2013) Packomania. http://www.packomania.com/
Stiefel E (1935–1936) Richtungsfelder und fernparallelismus in n-dimensionalem mannig faltigkeiten. Comment Math Helv 8:305–353
Stoica P, Babu P (2012) SPICE and LIKES: two hyperparameter-free methods for sparse-parameter estimation. Signal Process 92(8):1580–1590
Svanberg K (1987) The method of moving asymptotes-a new method for structural optimization. Int J Numer Methods Eng 24(2):359–373
Thoai NV, Tuy H (1980) Convergent algorithms for minimizing a concave function. Math Oper Res 5(4):556–566
Toh KC, Todd MJ, Tutuncu RH (1999) SDPT3—a MATLAB software package for semidefinite programming. Optim Methods Softw 11:545–581
Tutuncu RH, Toh KC, Todd MJ (2003) Solving semidfinite-quadratic-linear programs using SDPT3. Math Program 95(2):189–217
Tuy H (1983) On outer approximation methods for solving concave minimization problems. Acta Math Vietnam 8(2):3–34

Tuy H (1986) A general deterministic approach to global optimization via D.C. programming. N-Holl Math Stud 129:273–303

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

Wilson RB (1963) A simplicial algorithm for concave programming. PhD thesis, Gradaute School of Business Administration, Harvard University

Yamada S, Tanino T, Inuiguchi M (2000) Inner approximation method for a reverse convex programming problem. J Optim Theory Appl 107(2):355–389

Yu CNJ, Joachims T (2009) Learning structural SVMs with latent variables. In: Proceedings of the 26th annual international conference on machine learning, pp 1169–1176

Yuille AL, Rangarajan A (2003) The concave-convex procedure. Neural Comput 15(4):915–936

Zangwill WI (1969) Nonlinear programming: a unified approach. Prentice-Hall Inc, Englewood Cliffs

Zillober C (2001) Global convergence of a nonlinear programming method using convex optimization. Numer Algorithms 27(3):265–289