

NESTEROV'S SMOOTHING TECHNIQUE AND MINIMIZING DIFFERENCES OF CONVEX FUNCTIONS FOR HIERARCHICAL CLUSTERING

N. M. NAM¹, W. GEREMEW², S. REYNOLDS³ and T. TRAN⁴.

Abstract. A bilevel hierarchical clustering model is commonly used in designing optimal multicast networks. In this paper, we consider two different formulations of the bilevel hierarchical clustering problem, a discrete optimization problem which can be shown to be NP-hard. Our approach is to reformulate the problem as a continuous optimization problem by making some relaxations on the discreteness conditions. Then Nesterov's smoothing technique and a numerical algorithm for minimizing differences of convex functions called the DCA are applied to cope with the nonsmoothness and nonconvexity of the problem. Numerical examples are provided to illustrate our method.

Key words. DC programming, Nesterov's smoothing technique, hierarchical clustering, subgradient, Fenchel conjugate.

AMS subject classifications. 49J52, 49J53, 90C31

1 Introduction

Although convex optimization techniques and numerical algorithms have been the topics of extensive research for more than 50 years, solving large-scale optimization problems without the presence of convexity remains a challenge. This is a motivation to search for new optimization methods that are capable of handling broader classes of functions and sets where convexity is not assumed. One of the most successful approaches to go beyond convexity is to consider the class of functions representable as differences of two convex functions. Functions of this type are called *DC functions*, where DC stands for *difference of convex*. It was recognized early by P. Hartman [10] that the class of DC functions has many nice algebraic properties. For instance, this class of functions is closed under many operations usually considered in optimization such as taking linear combination, maximum, or product of a finite number of DC functions.

Given a linear space X , a *DC program* is an optimization problem in which the objective function $f: X \rightarrow \mathbb{R}$ can be represented as $f = g - h$, where $g, h: X \rightarrow \mathbb{R}$ are convex functions. This extension of convex programming enables us to take advantage of the available tools from convex analysis and optimization. At the same time, DC programming is sufficiently broad to use in solving many nonconvex optimization problems faced in recent applications. Another feature of DC programming is that it possesses a very nice duality theory; see [3] and the references therein. Although DC programming had been known to be important for many applications much earlier, the first algorithm for minimizing differences of convex

¹Fariborz Maseeh Department of Mathematics and Statistics, Portland State University, Portland, OR 97207, USA (mau.nam.nguyen@pdx.edu). Research of this author was partly supported by the National Science Foundation under grant #1411817.

²School of General Studies, Stockton University, Galloway, NJ 08205, USA (wondi.geremew@stockton.edu)

³Fariborz Maseeh Department of Mathematics and Statistics, Portland State University, Portland, OR 97207, USA (ser6@pdx.edu).

⁴Fariborz Maseeh Department of Mathematics and Statistics, Portland State University, Portland, OR 97207, USA (tuyen2@pdx.edu).

functions called the *DCA* was introduced by Tao and An in [3, 9]. The DCA is a simple but effective optimization scheme used extensively in DC programming and its applications.

Cluster analysis or *clustering* is one of the most important problems in many fields such as machine learning, pattern recognition, image analysis, data compression, and computer graphics. Given a finite number of data points in a metric space, a centroid-based clustering problem seeks a finite number of cluster centers with each data point assigned to the nearest cluster center in a way that a certain distance, a measure of dissimilarity among data points, is minimized. Since many kinds of data encountered in practical applications have nested structures, they are required to use multilevel hierarchical clustering which involves grouping a data set into a hierarchy of clusters. In this paper, we apply the mathematical optimization approach to the bilevel hierarchical clustering problem. In fact, using mathematical optimization in clustering is a very promising approach to overcome many disadvantages of the *k-mean algorithm* commonly used in clustering; see [1, 2, 5, 15] and the references therein. In particular, the DCA was successfully applied in [2] to a bilevel hierarchical clustering problem in which the distance measurement is defined by the squared Euclidean distance. Although the DCA in [2] provides an effective way to solve the bilevel hierarchical clustering in high dimensions, it has not been used to solve the original model defined by the Euclidean distance measurement proposed in [5] which is *not suitable for the resulting DCA* according to the authors of [2]. By applying Nesterov's smoothing technique and the DCA, we are able to solve the original model proposed in [5] in high dimensions.

The paper is organized as follows. In Section 2, we present basic definitions and tools of optimization that are used throughout the paper. In section 3, we study two models of bilevel hierarchical clustering problems, along with two new algorithms based on Nesterov's smoothing technique and the DCA. Numerical examples and conclusions are presented in Section 4 and Section 5, respectively.

2 Basic Definitions and Tools of Optimization

In this section, we present two main tools of optimization used to solve the bilevel hierarchical clustering problem: the DCA introduced by Pham Dinh Tao and Nesterov's smoothing technique.

We consider throughout the paper DC programming:

$$\text{minimize } f(x) := g(x) - h(x), x \in \mathbb{R}^n, \quad (2.1)$$

where $g: \mathbb{R}^n \rightarrow \mathbb{R}$ and $h: \mathbb{R}^n \rightarrow \mathbb{R}$ are convex functions. The function f in (2.1) is called a *DC function* and $g - h$ is called a *DC decomposition* of f .

Given a convex function $g: \mathbb{R}^n \rightarrow \mathbb{R}$, the *Fenchel conjugate* of g is defined by

$$g^*(y) := \sup\{\langle y, x \rangle - g(x) \mid x \in \mathbb{R}^n\}, y \in \mathbb{R}^n.$$

Note that $g^*: \mathbb{R}^n \rightarrow (-\infty, \infty]$ is also a convex function. In addition, $x \in \partial g^*(y)$ if and only if $y \in \partial g(x)$, where ∂ denotes the subdifferential operator in the sense of convex analysis; see, e.g., [19-21]

Let us present below the DCA introduced by Tao and An [3, 9] as applied to (2.1). Although the algorithm is used for nonconvex optimization problems, the convexity of the functions involved still plays a crucial role.

The DCA

```

INPUT:  $x_0 \in \mathbb{R}^n, N \in \mathbb{N}$ .
for  $k = 1, \dots, N$  do
    Find  $y_k \in \partial h(x_{k-1})$ .
    Find  $x_k \in \partial g^*(y_k)$ .
end for
OUTPUT:  $x_N$ .

```

Let us discuss below a convergence result of DC programming. A function $h: \mathbb{R}^n \rightarrow \mathbb{R}$ is called γ -convex ($\gamma \geq 0$) if the function defined by $k(x) := h(x) - \frac{\gamma}{2}\|x\|^2$, $x \in \mathbb{R}^n$, is convex. If there exists $\gamma > 0$ such that h is γ -convex, then h is called strongly convex. We say that an element $\bar{x} \in \mathbb{R}^n$ is a *critical point* of the function f defined by (2.1) if

$$\partial g(\bar{x}) \cap \partial h(\bar{x}) \neq \emptyset.$$

Obviously, in the case where both g and h are differentiable, \bar{x} is a critical point of f if and only if \bar{x} satisfies the Fermat rule $\nabla f(\bar{x}) = 0$. The theorem below provides a convergence result for the DCA. It can be derived directly from [9, Theorem 3.7].

Theorem 2.1 *Consider the function f defined by (2.1) and the sequence $\{x_k\}$ generated by the DCA. The following properties are valid:*

(i) *If g is γ_1 -convex and h is γ_2 -convex, then*

$$f(x_k) - f(x_{k+1}) \geq \frac{\gamma_1 + \gamma_2}{2} \|x_{k+1} - x_k\|^2 \text{ for all } k \in \mathbb{N}.$$

(ii) *The sequence $\{f(x_k)\}$ is monotone decreasing.*

(iii) *If f is bounded from below, g is γ_1 -convex and h is γ_2 -convex with $\gamma_1 + \gamma_2 > 0$, and $\{x_k\}$ is bounded, then every subsequential limit of the sequence $\{x_k\}$ is a critical point of f .*

Now we present a direct consequence of Nesterov's smoothing technique given in [16]. In the proposition below, $d(x; \Omega)$ denotes the Euclidean distance and $P(x; \Omega)$ denotes the Euclidean projection from a point x to a nonempty closed convex set Ω in \mathbb{R}^n .

Proposition 2.2 *Given any $a \in \mathbb{R}^n$ and $\mu > 0$, a Nesterov smoothing approximation of $\varphi(x) := \|x - a\|$ defined in \mathbb{R}^n has the representation*

$$\varphi_\mu(x) := \frac{1}{2\mu} \|x - a\|^2 - \frac{\mu}{2} \left[d\left(\frac{x - a}{\mu}; \mathbb{B}\right) \right]^2. \quad (2.2)$$

Moreover, $\nabla \varphi_\mu(x) = P\left(\frac{x - a}{\mu}; \mathbb{B}\right)$ and

$$\varphi_\mu(x) \leq \varphi(x) \leq \varphi_\mu(x) + \frac{\mu}{2},$$

where \mathbb{B} is the closed unit ball of \mathbb{R}^n .

3 The Bilevel Hierarchical Clustering Problem

Given a set of m points (nodes) a^1, a^2, \dots, a^m in \mathbb{R}^n , our goal is to decompose this set into k clusters. In each cluster, we would like to find a point x^i among the nodes and assign it as the center for this cluster with all points in the cluster connected to this center. Then we will find a *total center* x^* among the given points a^1, a^2, \dots, a^m , and all centers are connected to this total center. The goal is to minimize the *total transportation cost* in this tree computed by the sum of the distances from the total center to each center and from each center to the nodes in each cluster. This is a discrete optimization problem which can be shown to be *NP-hard*. We will solve this problem based on continuous optimization techniques.

3.1 The Bilevel Hierarchical Clustering: Model I

The difficulty in solving this hierarchical clustering problem lies in the fact that the centers and total center have to be among the nodes. We first relax this condition with the use of artificial centers x^1, x^2, \dots, x^k that could be anywhere in \mathbb{R}^n . Then we define the total center as the centroid of x^1, x^2, \dots, x^k given by

$$x^* := \frac{1}{k}(x^1 + x^2 + \dots + x^k).$$

The total cost of the tree is given by

$$\varphi(x^1, \dots, x^k) := \sum_{i=1}^m \min_{\ell=1, \dots, k} \|x^\ell - a^i\| + \sum_{\ell=1}^k \|x^\ell - x^*\|.$$

Note that here each a^i is assigned to its closest center. However, what we expect are the real centers, which can be approximated by trying to minimize the difference between the artificial centers and the real centers. To achieve this goal, define the function

$$\phi(x^1, \dots, x^k) := \sum_{\ell=1}^k \min_{i=1, \dots, m} \|x^\ell - a^i\|, x^1, \dots, x^k \in \mathbb{R}^n.$$

Observe that $\phi(x^1, \dots, x^k) = 0$ if and only if for every $\ell = 1, \dots, k$, there exists $i \in \{1, \dots, m\}$ such that $x^\ell = a^i$, which means that x^ℓ is a real node. Therefore, we consider the constrained minimization problem:

$$\text{minimize } \varphi(x^1, \dots, x^k) \text{ subject to } \phi(x^1, \dots, x^k) = 0.$$

This problem can be converted to an unconstrained minimization problem:

$$\text{minimize } f_\lambda(x^1, \dots, x^k) := \varphi(x^1, \dots, x^k) + \lambda \phi(x^1, \dots, x^k), x^1, \dots, x^k \in \mathbb{R}^n, \quad (3.1)$$

where $\lambda > 0$ is a penalty parameter. Similar to the situation with the clustering problem, this new problem is nonsmooth and nonconvex, which can be solved by smoothing techniques and the DCA. Note that a particular case of this model in two dimensions was

considered in [5] where the problem was solved using the derivative-free discrete gradient method established in [4], but this method is not suitable for large-scale settings in high dimensions. The DCA was used in [2] to solve a similar model with the squared Euclidean distance function used as the distance measurement. In this paper, the authors also addressed the difficulty of dealing with (3.1) using the DCA as *not suitable for the resulting DCA*. Nevertheless, we will show in what follows that the DCA is applicable to this model when combined with Nesterov's smoothing technique.

Note that the functions φ and ϕ in (3.1) belong to the class of DC functions with the following DC decompositions:

$$\begin{aligned}\varphi(x^1, \dots, x^k) &= \sum_{i=1}^m \left[\sum_{\ell=1}^k \|x^\ell - a^i\| - \max_{r=1, \dots, k} \sum_{\ell=1, \ell \neq r}^k \|x^\ell - a^i\| \right] + \sum_{\ell=1}^k \|x^\ell - x^*\| \\ &= \left[\sum_{i=1}^m \sum_{\ell=1}^k \|x^\ell - a^i\| + \sum_{\ell=1}^k \|x^\ell - x^*\| \right] - \sum_{i=1}^m \max_{r=1, \dots, k} \sum_{\ell=1, \ell \neq r}^k \|x^\ell - a^i\|, \\ \phi(x^1, \dots, x^k) &= \sum_{\ell=1}^k \sum_{i=1}^m \|x^\ell - a^i\| - \sum_{\ell=1}^k \max_{s=1, \dots, m} \sum_{i=1, i \neq s}^m \|x^\ell - a^i\|.\end{aligned}$$

It follows that the objective function f_λ in (3.1) has the DC decomposition:

$$\begin{aligned}f_\lambda(x^1, \dots, x^k) &= \left[(1 + \lambda) \sum_{\ell=1}^k \sum_{i=1}^m \|x^\ell - a^i\| + \sum_{\ell=1}^k \|x^\ell - x^*\| \right] \\ &\quad - \left[\sum_{i=1}^m \max_{r=1, \dots, k} \sum_{\ell=1, \ell \neq r}^k \|x^\ell - a^i\| + \lambda \sum_{\ell=1}^k \max_{s=1, \dots, m} \sum_{i=1, i \neq s}^m \|x^\ell - a^i\| \right].\end{aligned}$$

This DC decomposition is not suitable for applying the DCA because there is no closed form for a subgradient of the function g^* involved.

In the next step, we apply Nesterov's smoothing technique from Proposition 2.2 to approximate the objective function f_λ by a new DC function favorable for applying the DCA. To accomplish this goal, we simply replace each term of the form $\|x - a\|$ from the first part of $f_\lambda(x^1, \dots, x^k)$ (the *positive part*) by the smooth approximation (2.2), while keeping the second part (the *negative part*) the same. As a result, we obtain

$$\begin{aligned}f_{\lambda\mu}(x^1, \dots, x^k) &:= \frac{(1 + \lambda)\mu}{2} \sum_{i=1}^m \sum_{\ell=1}^k \left\| \frac{x^\ell - a^i}{\mu} \right\|^2 + \frac{\mu}{2} \sum_{\ell=1}^k \left\| \frac{x^\ell - x^*}{\mu} \right\|^2 \\ &\quad - \frac{(1 + \lambda)\mu}{2} \sum_{i=1}^m \sum_{\ell=1}^k \left[d \left(\frac{x^\ell - a^i}{\mu}; \mathbb{B} \right) \right]^2 - \frac{\mu}{2} \sum_{\ell=1}^k \left[d \left(\frac{x^\ell - x^*}{\mu}; \mathbb{B} \right) \right]^2 \\ &\quad - \sum_{i=1}^m \max_{r=1, \dots, k} \sum_{\ell=1, \ell \neq r}^k \|x^\ell - a^i\| - \lambda \sum_{\ell=1}^k \max_{s=1, \dots, m} \sum_{i=1, i \neq s}^m \|x^\ell - a^i\|.\end{aligned}$$

The original bilevel hierarchical clustering problem now can be solved using a DC program:

$$\text{minimize } f_{\lambda\mu}(x^1, \dots, x^k) = g_{\lambda\mu}(x^1, \dots, x^k) - h_{\lambda\mu}(x^1, \dots, x^k), \quad x^1, \dots, x^k \in \mathbb{R}^n.$$

In this formulation, $g_{\lambda\mu}$ and $h_{\lambda\mu}$ are convex functions on $(\mathbb{R}^n)^k$ defined by

$$\begin{aligned} g_{\lambda\mu}(x^1, \dots, x^k) &:= g_{\lambda\mu}^1(x^1, \dots, x^k) + g_{\lambda\mu}^2(x^1, \dots, x^k), \\ h_{\lambda\mu}(x^1, \dots, x^k) &:= h_{\lambda\mu}^1(x^1, \dots, x^k) + h_{\lambda\mu}^2(x^1, \dots, x^k) + h_{\lambda\mu}^3(x^1, \dots, x^k) + h_{\lambda\mu}^4(x^1, \dots, x^k), \end{aligned}$$

with their respective components defined as

$$\begin{aligned} g_{\lambda\mu}^1(x^1, \dots, x^k) &:= \frac{1+\lambda}{2\mu} \sum_{i=1}^m \sum_{\ell=1}^k \|x^\ell - a^i\|^2, \quad g_{\lambda\mu}^2(x^1, \dots, x^k) := \frac{1}{2\mu} \sum_{\ell=1}^k \|x^\ell - x^*\|^2, \\ h_{\lambda\mu}^1(x^1, \dots, x^k) &:= \frac{(1+\lambda)\mu}{2} \sum_{i=1}^m \sum_{\ell=1}^k \left[d\left(\frac{x^\ell - a^i}{\mu}; \mathbb{B}\right) \right]^2, \quad h_{\lambda\mu}^2(x^1, \dots, x^k) := \frac{\mu}{2} \sum_{\ell=1}^k \left[d\left(\frac{x^\ell - x^*}{\mu}; \mathbb{B}\right) \right]^2, \\ h_{\lambda\mu}^3(x^1, \dots, x^k) &:= \sum_{i=1}^m \max_{r=1, \dots, k} \sum_{\ell=1, \ell \neq r}^k \|x^\ell - a^i\|, \quad h_{\lambda\mu}^4(x^1, \dots, x^k) := \lambda \sum_{\ell=1}^k \max_{s=1, \dots, m} \sum_{i=1, i \neq s}^m \|x^\ell - a^i\|. \end{aligned}$$

To facilitate the gradient and subgradient calculations for the DCA, we introduce a *data matrix* \mathbf{A} and a *variable matrix* \mathbf{X} . The data \mathbf{A} is formed by putting each a^i , $i = 1, \dots, m$, in the i^{th} row, i.e.,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{pmatrix}.$$

Similarly, if x^1, \dots, x^k are the k cluster centers, then the variable \mathbf{X} is formed by putting each x^ℓ , $\ell = 1, \dots, k$, in the ℓ^{th} row, i.e.,

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ x_{k1} & x_{k2} & x_{k3} & \dots & x_{kn} \end{pmatrix}.$$

Then the variable matrix \mathbf{X} of the optimization problem belongs to $\mathbb{R}^{k \times n}$, the linear space of k by n real matrices equipped with the inner product $\langle \mathbf{X}, \mathbf{Y} \rangle := \text{trace}(\mathbf{X}^T \mathbf{Y})$. The *Frobenius norm* on $\mathbb{R}^{k \times n}$ is defined by

$$\|\mathbf{X}\|_F := \sqrt{\langle \mathbf{X}, \mathbf{X} \rangle} = \sqrt{\sum_{\ell=1}^k \langle x^\ell, x^\ell \rangle} = \sqrt{\sum_{\ell=1}^k \|x^\ell\|^2}.$$

Finally, we represent the average of the k cluster centers by x^* , i.e., $x^* := \frac{1}{k} \sum_{j=1}^k x^j$.

Gradient and Subgradient Calculations for the DCA

Let us start by computing the gradient of

$$g_{\lambda\mu}(\mathbf{X}) = g_{\lambda\mu}^1(\mathbf{X}) + g_{\lambda\mu}^2(\mathbf{X}).$$

Using the Frobenius norm, the function $g_{\lambda\mu}^1$ can equivalently be written as

$$\begin{aligned} g_{\lambda\mu}^1(\mathbf{X}) &= \frac{1+\lambda}{2\mu} \sum_{i=1}^m \sum_{\ell=1}^k \|x^\ell - a^i\|^2 \\ &= \frac{1+\lambda}{2\mu} \sum_{i=1}^m \sum_{\ell=1}^k \left[\|x^\ell\|^2 - 2\langle x^\ell, a^i \rangle + \|a^i\|^2 \right] \\ &= \frac{1+\lambda}{2\mu} \left[m\|\mathbf{X}\|_F^2 - 2\langle \mathbf{X}, \mathbf{E}_{km}\mathbf{A} \rangle + k\|\mathbf{A}\|_F^2 \right], \end{aligned}$$

where \mathbf{E}_{km} is a $k \times m$ matrix whose entries are all ones. Hence, one can see that $g_{\lambda\mu}^1$ is differentiable and its gradient is given by

$$\nabla g_{\lambda\mu}^1(\mathbf{X}) = \frac{1+\lambda}{\mu} [m\mathbf{X} - \mathbf{E}_{km}\mathbf{A}].$$

Similarly, $g_{\lambda\mu}^2$ can equivalently be written as

$$\begin{aligned} g_{\lambda\mu}^2(\mathbf{X}) &= \frac{1}{2\mu} \sum_{\ell=1}^k \|x^\ell - x^*\|^2 \\ &= \frac{1}{2\mu} \sum_{\ell=1}^k \left[\|x^\ell\|^2 - 2\langle x^\ell, x^* \rangle + \|x^*\|^2 \right] \\ &= \frac{1}{2\mu} \left[\|\mathbf{X}\|_F^2 - \frac{2}{k} \langle \mathbf{X}, \mathbf{E}_{kk}\mathbf{X} \rangle + \frac{1}{k} \langle \mathbf{X}, \mathbf{E}_{kk}\mathbf{X} \rangle \right] \\ &= \frac{1}{2\mu} \left[\|\mathbf{X}\|_F^2 - \frac{1}{k} \langle \mathbf{X}, \mathbf{E}_{kk}\mathbf{X} \rangle \right], \end{aligned}$$

where \mathbf{E}_{kk} is a $k \times k$ matrix whose entries are all ones. Hence, $g_{\lambda\mu}^2$ is differentiable and its gradient is given by

$$\nabla g_{\lambda\mu}^2(\mathbf{X}) = \frac{1}{\mu} \left[\mathbf{X} - \frac{1}{k} \mathbf{E}_{kk}\mathbf{X} \right].$$

Since $g_{\lambda\mu}(\mathbf{X}) = g_{\lambda\mu}^1(\mathbf{X}) + g_{\lambda\mu}^2(\mathbf{X})$, its gradient can be computed by

$$\begin{aligned} \nabla g_{\lambda\mu}(\mathbf{X}) &= \nabla g_{\lambda\mu}^1(\mathbf{X}) + \nabla g_{\lambda\mu}^2(\mathbf{X}) \\ &= \frac{1+\lambda}{\mu} [m\mathbf{X} - \mathbf{E}_{km}\mathbf{A}] + \frac{1}{\mu} \left[\mathbf{X} - \frac{1}{k} \mathbf{E}_{kk}\mathbf{X} \right] \\ &= \frac{1}{\mu} \left[(1+\lambda)m\mathbf{X} - (1+\lambda)\mathbf{E}_{km}\mathbf{A} + \mathbf{X} - \frac{1}{k} \mathbf{E}_{kk}\mathbf{X} \right] \\ &= \frac{1}{\mu} \left[\left[(1+\lambda)m + 1 \right] \mathbf{I}_{kk} - \frac{1}{k} \mathbf{E}_{kk} \right] \mathbf{X} - (1+\lambda)\mathbf{E}_{km}\mathbf{A}. \end{aligned}$$

Therefore,

$$\nabla g_{\lambda\mu}(\mathbf{X}) = \frac{1}{\mu} \left[\left((1+\lambda)m + 1 \right) \mathbf{I}_{kk} - \mathbf{J} \right] \mathbf{X} - (1+\lambda)\mathbf{S}, \quad \text{where } \mathbf{J} = \frac{1}{k} \mathbf{E}_{kk}, \text{ and } \mathbf{S} = \mathbf{E}_{km}\mathbf{A}.$$

Our goal now is to compute $\nabla g^*(Y)$, which can be accomplished by the relation

$$\mathbf{X} = \nabla g^*(\mathbf{Y}) \text{ if and only if } \mathbf{Y} = \nabla g(\mathbf{X}).$$

The latter can equivalently be written as

$$[[1 + (1 + \lambda)m] \mathbf{I}_{kk} - \mathbf{J}] \mathbf{X} = [(1 + \lambda)\mathbf{S} + \mu\mathbf{Y}].$$

Then with some algebraic manipulation we can show that

$$\nabla g^*(\mathbf{Y}) = \mathbf{X} = \left[\frac{1}{1 + (1 + \lambda)m} \mathbf{I}_{kk} + \frac{1}{[1 + (1 + \lambda)m](1 + \lambda)m} \mathbf{J} \right] [(1 + \lambda)\mathbf{S} + \mu\mathbf{Y}].$$

Next, we will demonstrate in more detail the techniques we used in finding a subgradient for the convex function $h_{\lambda\mu}$. Recall that $h_{\lambda\mu}$ is defined by

$$h_{\lambda\mu}(\mathbf{X}) = \sum_{i=1}^4 h_{\lambda\mu}^i(\mathbf{X}).$$

We will start with the function $h_{\lambda\mu}^1$ given by

$$h_{\lambda\mu}^1(\mathbf{X}) = \frac{(1 + \lambda)\mu}{2} \sum_{i=1}^m \sum_{\ell=1}^k \left[d \left(\frac{x^\ell - a^i}{\mu}; \mathbb{B} \right) \right]^2.$$

From its representation, one can see that $h_{\lambda\mu}^1$ is differentiable, and hence its subgradient coincides with its gradient, that can be computed by the partial derivatives with respect to x^1, \dots, x^k , i.e.,

$$\frac{\partial h_{\lambda\mu}^1}{\partial x^\ell}(\mathbf{X}) = (1 + \lambda) \sum_{i=1}^m \left[\frac{x^\ell - a^i}{\mu} - P \left(\frac{x^\ell - a^i}{\mu}; \mathbb{B} \right) \right].$$

Thus, for $\ell = 1, 2, \dots, k$, $\nabla h_{\lambda\mu}^1(\mathbf{X})$ is the $k \times n$ matrix \mathbf{U} whose ℓ^{th} row is $\frac{\partial h_{\lambda\mu}^1}{\partial x^\ell}(\mathbf{X})$.

Similarly, one can see that the function $h_{\lambda\mu}^2$ given by

$$h_{\lambda\mu}^2(\mathbf{X}) = \frac{\mu}{2} \sum_{\ell=1}^k \left[d \left(\frac{x^\ell - x^*}{\mu}; \mathbb{B} \right) \right]^2$$

is differentiable with its partial derivatives computed by

$$\frac{\partial h_{\lambda\mu}^2}{\partial x^\ell}(\mathbf{X}) = \left[\frac{x^\ell - x^*}{\mu} - P \left(\frac{x^\ell - x^*}{\mu}; \mathbb{B} \right) \right] - \frac{1}{k} \sum_{j=1}^k \left[\frac{x^j - x^*}{\mu} - P \left(\frac{x^j - x^*}{\mu}; \mathbb{B} \right) \right].$$

Hence, for $\ell = 1, 2, \dots, k$, $\nabla h_{\lambda\mu}^2(\mathbf{X})$ is the $k \times n$ matrix \mathbf{V} whose ℓ^{th} row is $\frac{\partial h_{\lambda\mu}^2}{\partial x^\ell}(\mathbf{X})$.

Unlike $h_{\lambda\mu}^1$ and $h_{\lambda\mu}^2$, the convex functions $h_{\lambda\mu}^3$ and $h_{\lambda\mu}^4$ are not differentiable, but both can be written as a finite sum of the maximum of a finite number of convex functions. Let us compute a subgradient of $h_{\lambda\mu}^3$ as an example. We have

$$h_{\lambda\mu}^3(\mathbf{X}) = \sum_{i=1}^m \max_{r=1, \dots, k} \sum_{\ell=1, \ell \neq r}^k \|x^\ell - a^i\| = \sum_{i=1}^m \gamma_i(\mathbf{X}),$$

where, for $i = 1, \dots, m$,

$$\gamma_i(\mathbf{X}) := \max \left\{ \gamma_{ir}(\mathbf{X}) = \sum_{\ell=1, \ell \neq r}^k \|x^\ell - a^i\|, \quad r = 1, \dots, k \right\}.$$

Then, for each $i = 1, \dots, m$, we find $\mathbf{W}_i \in \partial\gamma_i(\mathbf{X})$ according to the subdifferential rule for the maximum of convex functions. Then define $\mathbf{W} := \sum_{i=1}^m \mathbf{W}_i$ to get a subgradient of the function $h_{\lambda\mu}^3$ at \mathbf{X} by the subdifferential sum rule. To accomplish this goal, we first choose an index r^* from the index set $\{1, \dots, k\}$ such that

$$\gamma_i(\mathbf{X}) = \gamma_{ir^*}(\mathbf{X}) = \sum_{\ell=1, \ell \neq r^*}^k \|x^\ell - a^i\|.$$

Using the familiar subdifferential formula of the Euclidean norm function, the ℓ^{th} row w_i^ℓ for $\ell \neq r^*$ of the matrix \mathbf{W}_i is determined as follows

$$w_i^\ell := \begin{cases} \frac{x^\ell - a^i}{\|x^\ell - a^i\|_2} & \text{if } x^\ell \neq a^i, \\ u \in \mathbb{B} & \text{if } x^\ell = a^i. \end{cases}$$

The r^{*th} row of the matrix \mathbf{W}_i is $w_i^{r^*} := 0$.

The procedure for calculating a subgradient of the function $h_{\lambda\mu}^4$ given by

$$h_{\lambda\mu}^4(x^1, \dots, x^k) = \lambda \sum_{\ell=1}^k \max_{s=1, \dots, m} \sum_{i=1, i \neq s}^m \|x^\ell - a^i\|,$$

is very similar to what we just demonstrated for $h_{\lambda\mu}^3$.

At this point, we are ready to give a new DCA based algorithm for Model I.

Algorithm 1.

```

INPUT:  $\mathbf{X}_0, \lambda_0, \mu_0, N \in \mathbb{N}$ .
while stopping criteria  $(\lambda, \mu) = \text{false}$  do
  for  $k = 1, 2, 3, \dots, N$  do
    Find  $\mathbf{Y}_k \in \partial h_{1\lambda\mu}(\mathbf{X}_{k-1})$ .
    Find  $\mathbf{X}_k \in \partial g_{1\lambda\mu}^*(\mathbf{Y}_k)$ .
  end for
  update  $\lambda$  and  $\mu$ .
end while
OUTPUT:  $\mathbf{X}_N$ .

```

3.2 The Bilevel Hierarchical Clustering: Model II

In this section, we introduce the second model to solve the bilevel hierarchical clustering problem. In this model, we use an additional variable x^{k+1} to denote the total center. At

first we allow the total center x^{k+1} to be a free point in \mathbb{R}^n , the same as the k cluster centers. Then the total cost of the tree is given by

$$\varphi(x^1, \dots, x^{k+1}) := \sum_{i=1}^m \min_{\ell=1, \dots, k} \|x^\ell - a^i\| + \sum_{\ell=1}^k \|x^\ell - x^{k+1}\|, x^1, \dots, x^{k+1} \in \mathbb{R}^n.$$

To force the $k+1$ centers to be chosen from the given nodes (or to make them as close to the nodes as possible), we set the constraint

$$\phi(x^1, \dots, x^{k+1}) := \sum_{\ell=1}^{k+1} \min_{i=1, \dots, m} \|x^\ell - a^i\| = 0.$$

Our goal is to solve the optimization problem

$$\begin{aligned} & \text{minimize } \varphi(x^1, \dots, x^{k+1}) \\ & \text{subject to } \phi(x^1, \dots, x^{k+1}), x^1, \dots, x^{k+1} \in \mathbb{R}^n. \end{aligned}$$

Similar to the first model, this problem formulation can be converted to an unconstrained minimization problem involving a penalty parameter $\lambda > 0$:

$$\text{minimize } f_\lambda(x^1, \dots, x^{k+1}) := \varphi(x^1, \dots, x^{k+1}) + \lambda \phi(x^1, \dots, x^{k+1}), x^1, \dots, x^{k+1} \in \mathbb{R}^n. \quad (3.2)$$

Next, we apply Nesterov's smoothing technique to get an approximation of the objective function f given in (3.2) which involves two parameter $\lambda > 0$ and $\mu > 0$:

$$\begin{aligned} f_{\lambda\mu}(\mathbf{X}) &:= \frac{1+\lambda}{2\mu} \sum_{i=1}^m \sum_{\ell=1}^{k+1} \|x^\ell - a^i\|^2 + \frac{1}{2\mu} \sum_{\ell=1}^k \|x^\ell - x^{k+1}\|^2 - \frac{1}{2\mu} \sum_{i=1}^m \|x^{k+1} - a^i\|^2 \\ & - \frac{\lambda\mu}{2} \sum_{i=1}^m \left[d\left(\frac{x^{k+1} - a^i}{\mu}; \mathbb{B}\right) \right]^2 - \frac{(1+\lambda)\mu}{2} \sum_{i=1}^m \sum_{\ell=1}^k \left[d\left(\frac{x^\ell - a^i}{\mu}; \mathbb{B}\right) \right]^2 \\ & - \frac{\mu}{2} \sum_{\ell=1}^k \left[d\left(\frac{x^\ell - x^{k+1}}{\mu}; \mathbb{B}\right) \right]^2 - \sum_{i=1}^m \max_{r=1, \dots, k} \sum_{\ell=1, \ell \neq r}^k \|x^\ell - a^i\| - \lambda \sum_{\ell=1}^{k+1} \max_{s=1, \dots, m} \sum_{i=1, i \neq s}^m \|x^\ell - a^i\|. \end{aligned}$$

As we will show in what follows, it is convenient to apply the DCA to minimize the function $f_{\lambda\mu}$. This function can be represented as the differences of two convex functions defined on $\mathbb{R}^{(k+1) \times n}$ using a variable X whose i^{th} row is x^i for $i = 1, \dots, k+1$:

$$f_{\lambda\mu}(\mathbf{X}) = g_{\lambda\mu}(\mathbf{X}) - h_{\lambda\mu}(\mathbf{X}), \quad \mathbf{X} \in \mathbb{R}^{(k+1) \times n}.$$

In this formulation, $g_{\lambda\mu}$ and $h_{\lambda\mu}$ are convex functions defined on $\mathbb{R}^{(k+1) \times n}$ by

$$g_{\lambda\mu}(\mathbf{X}) := g_{\lambda\mu}^1(\mathbf{X}) + g_{\lambda\mu}^2(\mathbf{X})$$

and

$$h_{\lambda\mu}(\mathbf{X}) := h_{\lambda\mu}^1(\mathbf{X}) + h_{\lambda\mu}^2(\mathbf{X}) + h_{\lambda\mu}^3(\mathbf{X}) + h_{\lambda\mu}^4(\mathbf{X}) + h_{\lambda\mu}^5(\mathbf{X}) + h_{\lambda\mu}^6(\mathbf{X}),$$

with their respective components given by

$$\begin{aligned}
g_{\lambda\mu}^1(\mathbf{X}) &:= \frac{1+\lambda}{2\mu} \sum_{i=1}^m \sum_{\ell=1}^{k+1} \|x^\ell - a^i\|^2, \quad g_{\lambda\mu}^2(\mathbf{X}) := \frac{1}{2\mu} \sum_{\ell=1}^k \|x^\ell - x^{k+1}\|^2, \\
h_{\lambda\mu}^1(\mathbf{X}) &:= \frac{1}{2\mu} \sum_{i=1}^m \|x^{k+1} - a^i\|^2, \quad h_{\lambda\mu}^2(\mathbf{X}) := \frac{\lambda\mu}{2} \sum_{i=1}^m \left[d\left(\frac{x^{k+1} - a^i}{\mu}; \mathbb{B}\right) \right]^2, \\
h_{\lambda\mu}^3(\mathbf{X}) &:= \frac{(1+\lambda)\mu}{2} \sum_{i=1}^m \sum_{\ell=1}^k \left[d\left(\frac{x^\ell - a^i}{\mu}; \mathbb{B}\right) \right]^2, \quad h_{\lambda\mu}^4(\mathbf{X}) := \frac{\mu}{2} \sum_{\ell=1}^k \left[d\left(\frac{x^\ell - x^{k+1}}{\mu}; \mathbb{B}\right) \right]^2, \\
h_{\lambda\mu}^5(\mathbf{X}) &:= \sum_{i=1}^m \max_{r=1, \dots, k} \sum_{\ell=1, \ell \neq r}^k \|x^\ell - a^i\|, \quad h_{\lambda\mu}^6(\mathbf{X}) := \lambda \sum_{\ell=1}^{k+1} \max_{s=1, \dots, m} \sum_{i=1, i \neq s}^m \|x^\ell - a^i\|.
\end{aligned}$$

Gradient and Subgradient Calculations for the DCA

Let us start by computing the gradient of the first part of the DC decomposition, i.e.,

$$g_{\lambda\mu}(\mathbf{X}) = g_{\lambda\mu}^1(\mathbf{X}) + g_{\lambda\mu}^2(\mathbf{X}).$$

By applying similar techniques used in computing gradients/subgradients for Model I, $\nabla g_{\lambda\mu}^1(\mathbf{X})$ can be written as

$$\nabla g_{\lambda\mu}^1(\mathbf{X}) = \frac{1+\lambda}{\mu} [m\mathbf{X} - \mathbf{EA}],$$

where \mathbf{E} is the $(k+1) \times m$ matrix whose entries are all ones. Similarly, $\nabla g_{\lambda\mu}^2(\mathbf{X})$ can also be written as

$$\nabla g_{\lambda\mu}^2(\mathbf{X}) = \frac{1}{\mu} [\mathbf{I} + \mathbf{T}] \mathbf{X},$$

where \mathbf{I} is the $(k+1) \times (k+1)$ identity matrix, and \mathbf{T} is the $(k+1) \times (k+1)$ matrix whose entries are all zeros except its $(k+1)^{th}$ row and $(k+1)^{th}$ column, which both are filled by the vector $(-1, -1, -1, \dots, -1, k-1)$.

It follows that

$$\begin{aligned}
\nabla g_{\lambda\mu}(\mathbf{X}) &= \nabla g_{\lambda\mu}^1(\mathbf{X}) + \nabla g_{\lambda\mu}^2(\mathbf{X}) \\
&= \frac{1+\lambda}{\mu} [m\mathbf{X} - \mathbf{EA}] + \frac{1}{\mu} [\mathbf{I} + \mathbf{T}] \mathbf{X} \\
&= \frac{1}{\mu} [c_1\mathbf{I} + \mathbf{T}] \mathbf{X} - \frac{1+\lambda}{\mu} \mathbf{EA},
\end{aligned}$$

where $c_1 = 1 + (1+\lambda)m$. Our goal now is to compute $\nabla g_{\lambda\mu}^*(\mathbf{Y})$, which can be accomplished by the relation

$$\mathbf{X} = \nabla g_{\lambda\mu}^*(\mathbf{Y}) \text{ if and only if } \mathbf{Y} = \nabla g_{\lambda\mu}(\mathbf{X}).$$

The latter can be equivalently written as

$$[c_1\mathbf{I} + \mathbf{T}] \mathbf{X} = (1+\lambda)\mathbf{EA} + \mu\mathbf{Y},$$

whose solutions can be explicitly computed by its ℓ^{th} row for $\ell = 1, \dots, k+1$:

$$x^\ell = \frac{[(1+\lambda)\mathbf{EA} + \mu\mathbf{Y}]_\ell + x^{k+1}}{c_1} \text{ for } \ell = 1, \dots, k,$$

$$x^{k+1} = \frac{c_1 [(1+\lambda)\mathbf{EA} + \mu\mathbf{Y}]_{k+1} + \sum_{\ell=1}^k [(1+\lambda)\mathbf{EA} + \mu\mathbf{Y}]_\ell}{(c_1 + k)(c_1 - 1)}.$$

In the representation

$$h_{\lambda\mu}(\mathbf{X}) = h_{\lambda\mu}^1(\mathbf{X}) + h_{\lambda\mu}^2(\mathbf{X}) + h_{\lambda\mu}^3(\mathbf{X}) + h_{\lambda\mu}^4(\mathbf{X}) + h_{\lambda\mu}^5(\mathbf{X}) + h_{\lambda\mu}^6(\mathbf{X}),$$

the convex functions $h_{\lambda\mu}^1$, $h_{\lambda\mu}^2$, $h_{\lambda\mu}^3$, and $h_{\lambda\mu}^4$ are differentiable. The *partial derivatives* of $h_{\lambda\mu}^1$ are given by

$$\frac{\partial h_{\lambda\mu}^1}{\partial x^\ell}(\mathbf{X}) = 0 \text{ for } \ell = 1, \dots, k,$$

$$\frac{\partial h_{\lambda\mu}^1}{\partial x^{k+1}}(\mathbf{X}) = \frac{1}{\mu} \sum_{i=1}^m (x^{k+1} - a^i) = \frac{1}{\mu} \left[mx^{k+1} - \sum_{i=1}^m \mathbf{A}_i \right].$$

Then $\nabla h_{\lambda\mu}^1(\mathbf{X})$ is the $(k+1) \times n$ matrix \mathbf{L} whose ℓ^{th} row is $\frac{\partial h_{\lambda\mu}^1}{\partial x^\ell}(\mathbf{X})$ for $\ell = 1, \dots, k+1$. Similarly, the partial derivatives of $h_{\lambda\mu}^2$ are given by

$$\frac{\partial h_{\lambda\mu}^2}{\partial x^\ell}(\mathbf{X}) = 0 \text{ for } \ell = 1, \dots, k,$$

$$\frac{\partial h_{\lambda\mu}^2}{\partial x^{k+1}}(\mathbf{X}) = \lambda \sum_{i=1}^m \left[\frac{x^{k+1} - a^i}{\mu} - P \left(\frac{x^{k+1} - a^i}{\mu}; \mathbb{B} \right) \right].$$

Then $\nabla h_{\lambda\mu}^2(\mathbf{X})$ is the $(k+1) \times n$ matrix \mathbf{M} whose ℓ^{th} row is $\frac{\partial h_{\lambda\mu}^2}{\partial x^\ell}(\mathbf{X})$, for $\ell = 1, \dots, k+1$. Now we compute the gradient of $h_{\lambda\mu}^3$. We have

$$\frac{\partial h_{\lambda\mu}^3}{\partial x^\ell}(\mathbf{X}) = (1+\lambda) \sum_{i=1}^m \left[\frac{x^\ell - a^i}{\mu} - P \left(\frac{x^\ell - a^i}{\mu}; \mathbb{B} \right) \right] \text{ for } \ell = 1, \dots, k,$$

$$\frac{\partial h_{\lambda\mu}^3}{\partial x^{k+1}}(\mathbf{X}) = 0.$$

Thus, $\nabla h_{\lambda\mu}^3(\mathbf{X})$ is the $(k+1) \times n$ matrix \mathbf{U} whose l^{th} row is $\frac{\partial h_{\lambda\mu}^3}{\partial x^\ell}(\mathbf{X})$ for $\ell = 1, \dots, k+1$.

Let us compute the gradient of $h_{\lambda\mu}^4$. We have

$$\frac{\partial h_{\lambda\mu}^4}{\partial x^\ell}(\mathbf{X}) = \left[\frac{x^\ell - x^{k+1}}{\mu} - P \left(\frac{x^\ell - x^{k+1}}{\mu}; \mathbb{B} \right) \right] \text{ for } \ell = 1, \dots, k,$$

$$\frac{\partial h_{\lambda\mu}^4}{\partial x^{k+1}}(\mathbf{X}) = - \sum_{\ell=1}^k \left[\frac{x^\ell - x^{k+1}}{\mu} - P \left(\frac{x^\ell - x^{k+1}}{\mu}; \mathbb{B} \right) \right].$$

Similarly, $\nabla h_{\lambda\mu}^4(\mathbf{X})$ is the $(k+1) \times n$ matrix V whose ℓ^{th} row is filled with $\frac{\partial h_{\lambda\mu}^4}{\partial x^\ell}(\mathbf{X})$, for $\ell = 1, \dots, k+1$.

The procedure for computing subgradients of the last two nondifferentiable components, $h_{\lambda\mu}^5$ and $h_{\lambda\mu}^6$, is similar to the procedure we described for computing subgradients of $h_{\lambda\mu}^3$ in Model I.

The following is a DCA based algorithm for Model II.

Algorithm 2.

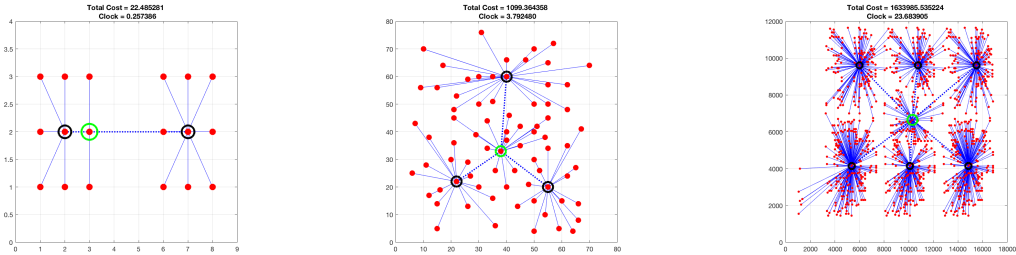
```

INPUT:  $\mathbf{X}_0, \lambda_0, \mu_0, N \in \mathbb{N}$ .
while stopping criteria  $(\lambda, \mu) = \text{false}$  do
  for  $k = 1, 2, 3, \dots, N$  do
    Find  $\mathbf{Y}_k \in \partial h_{\lambda\mu}(X_{k-1})$ .
    Find  $\mathbf{X}_k \in \partial g_{\lambda\mu}^*(Y_k)$ .
  end for
  update  $\lambda$  and  $\mu$ .
end while
OUTPUT:  $\mathbf{X}_N$ .

```

4 Numerical Experiments

We use MATLAB to code our algorithms and perform numerical experiments on a MacBook Pro with 2.2 GHz Intel Core i7 Processor, and 16 GB 1600 MHz DDR3 Memory. For our numerical experiments, we use three data sets: one artificial data set with 18 data points in \mathbb{R}^2 (see Figure 1a), the EIL76 and the PR1002 from [17] (see Figure 1b and Figure 1c), respectively.



(a) 18 Data Points, 2 Centers (b) 76 Data Points, 3 Centers (c) 1002 Data Points, 6 Centers

Figure 1: Plots of the three test data sets

The two MATLAB codes used to implement our two algorithms have two major parts: an *outer* loop for updating the penalty and the smoothing parameters and an *inner* loop for updating the cluster centers. The penalty parameter λ and the smoothing parameter μ are updated as follows. Choosing $\mu_0 > 0$, $\lambda_0 > 0$ and $\sigma_1 > 1$, $\sigma_2 \in (0, 1)$, we update $\lambda_{i+1} = \sigma_1 \lambda_i$ and $\mu_{i+1} = \sigma_2 \mu_i$ for $i \geq 0$ after each outer loop. To choose σ_1 and σ_2 , we first

let N be the number of outer iterations and choose $\lambda_0, \lambda_{\max}$ as the initial and final values of λ , respectively, and similarly for μ_0, μ_{\min} . Then choose growth/decay parameters according to $\sigma_1 = (\lambda_{\max}/\lambda_0)^{1/N}$ and $\sigma_2 = (\mu_{\min}/\mu_0)^{1/N}$.

By trial and error we find that the values chosen for $\lambda_0, \lambda_{\max}, \mu_0$, and μ_{\min} in large part determine the performance of the two algorithms for each data set. Intuitively, we see that very large values of λ will over-penalize the distance between an artificial center and its nearest data node and may prevent the algorithm from clustering properly. We therefore use $\lambda_0 \leq 1 \ll \lambda_{\max}$ so that the algorithm has a chance to cluster the data before the penalty parameter takes effect. Similarly, we choose $\mu_{\min} \ll 1 \leq \mu_0$.

We select the starting center \mathbf{X}_0 at a certain radius, $\gamma \text{rad}(\mathbf{A})$, from the median point, $\text{median}(\mathbf{A})$, of the entire data set, i.e.,

$$\mathbf{X}_0 = \text{median}(\mathbf{A}) + \gamma \text{rad}(\mathbf{A}) \mathbf{U},$$

where $\text{rad}(\mathbf{A}) := \max\{\|a^i - \text{median}(\mathbf{A})\| \mid a^i \in \mathbf{A}\}$, γ is a randomly chosen real number from the standard uniform distribution on the open interval $(0,1)$, \mathbf{U} is a $k \times n$ matrix whose k rows are randomly generated unit vectors in \mathbb{R}^n , and the sum is in the sense of adding a vector to each row of a matrix.

As showed in Tables 1, 2, and 3, both algorithms identify the optimal solutions with reasonable amount of time for both DS18 and EIL76 with two and three cluster centers, respectively. To get a good starting point which yields a better estimate of the optimal value for bigger data sets such as PR1002, we use a method called *radial search* described as follows. Given initial radius $r_0 > 0$ and $m \in \mathbb{N}$, set $\gamma = ir_0$ for $i = 1, \dots, m$. Then we test the algorithm with different starting points given by $\mathbf{X}_0(i) = \text{median}(\mathbf{A}) + ir_0(\text{rad}(\mathbf{A})\mathbf{U})$ for $i = 1, \dots, m$. Figure 2 shows the result of the method applied to PR1002 with six cluster centers, where the y -axis represents the optimal value returned by ALG1 with different starting points $\mathbf{X}_0(i)$, as represented on the x -axis.

$$\mu_0 = 5.70, \lambda_0 = 0.001, \sigma_1 = 7500, \sigma_2 = 0.5$$

	Cost1	Cost2	Time1	Time2	Iter1	Iter2	k	m	n
ADS18	22.4853	22.4853	0.0690125	0.0853712	124	124	2	18	2
ADS18	22.4853	22.4853	0.0678142	0.0902829	124	124	2	18	2
ADS18	22.4853	22.4853	0.0694841	0.0970158	124	124	2	18	2

Table 1: Results for the 18 points artificial data set.

$$\mu_0 = 100, \lambda_0 = 10^{-6}, \sigma_1 = 1, \sigma_2 = 0.5$$

	Cost1	Cost2	Time1	Time2	Iter1	Iter2	k	m	n
EIL76	1125.48	1107.47	1.6126	1.86261	540	540	3	76	2
EIL76	1099.36	1099.36	1.52001	1.81243	540	540	3	76	2
EIL76	1099.36	1099.36	1.52914	1.86536	540	540	3	76	2

Table 2: Results for EIL76 data set.

$$\mu_0 = 1950, \lambda_0 = 10^{-6}, \sigma_1 = 7500, \sigma_2 = 0.5$$

	Cost1	Cost2	Time1	Time2	Iter1	Iter2	k	m	n
PR1002	1.63399e+06	1.63399e+06	22.1578	25.3651	330	330	6	1002	2
PR1002	1.63399e+06	1.63399e+06	22.2978	25.5373	330	330	6	1002	2
PR1002	1.63399e+06	1.63399e+06	23.5394	26.0762	330	330	6	1002	2

Table 3: Results for PR1002 data set.

For comparison purposes, both Cost1 and Cost2 are computed by the same way. First, we systematically reassign the k cluster centers returned by the respective algorithms by k real nodes that are close to them, i.e., for $\ell = 1, \dots, k$

$$\bar{x}^\ell = \operatorname{argmin}\{\|x^\ell - a^i\| \mid a^i \in A\}.$$

Then the total center x^* will be a real node, from the remaining nodes, whose sum of distances from the k reassigned centers is the minimal, i.e.,

$$x^* := \operatorname{argmin}\left\{\sum_{\ell=1}^k \|\bar{x}^\ell - a^i\| \mid a^i \in A\right\}.$$

The total cost is computed by adding the distance of each real node to its closest center (including the total center), and the distances of the total center from the k cluster centers, i.e.,

$$\text{Cost} := \sum_{i=1}^m \min_{\ell=1, \dots, k+1} \|\bar{x}^\ell - a^i\| + \sum_{\ell=1}^k \|\bar{x}^\ell - x^{k+1}\|, \text{ where } x^{k+1} = x^*.$$

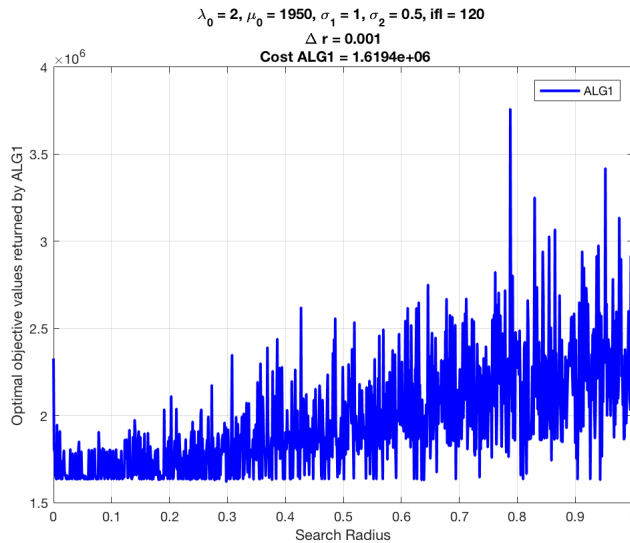


Figure 2: PR1002, The 1002 City Problem, with 6 Cluster Centers

5 Conclusions

In our numerical experiments, one of the major challenge we face is how to choose optimal parameters for our algorithms. We observe that parameter selection is the decisive factor in terms of accuracy and speed of convergence of our proposed algorithms. The performance of the proposed algorithms highly depends on the initial values set to the penalty and smoothing parameters, λ_0 and μ_0 ; and their respective growth/decay factors σ_1 and σ_2 . A better guidance of selecting the optimal parameter than the methods we suggest in this paper will be addressed in or future research.

References

- [1] L. T. H. An, M. T. Belghiti, P. D. Tao: A new efficient algorithm based on DC programming and DCA for clustering. *J. Glob. Optim.*, **27**, 503–608 (2007)
- [2] L. T. H. An, and L. H. Minh, Optimization based DC programming and DCA for hierarchical clustering. *European J. Oper. Res.* **183** (2007), 1067–1085.
- [3] L. T. H. An, and P. D. Tao, Convex analysis approach to D.C. programming: Theory, algorithms and applications. *Acta Math. Vietnam.* **22** (1997), 289–355.
- [4] A. M. Bagirov, Derivative-free methods for unconstrained nonsmooth optimization and its numerical analysis. *Investigacao Operacional.* **19** (1999), 75–93.
- [5] A. Bagirov, Long Jia, I. Ouyesi, and A.M. Rubinov, Optimization based clustering algorithms in Multicast group hierarchies, in: Proceedings of the Australian Telecommunications, Networks and Applications Conference (ATNAC), 2003, Melbourne Australia (published on CD, ISBN 0-646-42229-4).
- [6] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces* Springer, New York, 2011.
- [7] J. M. Borwein and A. S. Lewis, *Convex Analysis and Nonlinear Optimization*, 2nd edition, Springer, New York, 2006.
- [8] R. I. Boş, *Conjugate Duality in Convex Optimization*, Springer, Berlin, 2010.
- [9] T. Pham Dinh and H. A. Le Thi, A d.c. optimization algorithm for solving the trust-region subproblem, *SIAM J. Optim.* **8** (1998), 476–505.
- [10] P. Hartman, On functions representable as a difference of convex functions. *Pacific J. Math.* **9**, (1959), 707–713.
- [11] J.-B. Hiriart-Urruty, C. Lemaréchal, *Convex Analysis and Minimization Algorithms I, II*, Springer, Berlin, 1993.
- [12] J. B. Hiriart-Urruty, Generalized differentiability, duality and optimization for problems dealing with differences of convex functions. *Lecture Note in Economics and Math. Systems.* **256** (1985), 37–70.
- [13] B. S. Mordukhovich, *Variational Analysis and Generalized Differentiation, I: Basic Theory, II: Applications*, Springer, Berlin, 2006.
- [14] B. S. Mordukhovich and N. M. Nam, *An Easy Path to Convex Analysis and Applications*, Morgan & Claypool Publishers, San Rafael, CA, 2014.

- [15] N. M. Nam, R.B. Rector, D. Giles: Minimizing Differences of Convex Functions with Applications to Facility Location and Clustering, submitted.
- [16] Y. Nesterov: Smooth minimization of non-smooth functions. *Math. Program.* **103**, 127–152 (2005)
- [17] G. Reinelt, TSPLIB: A Traveling Salesman Problem Library. *ORSA Journal of Computing.* **3** (1991), 376–384.
- [18] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.
- [19] R. T. Rockafellar, *Conjugate Duality and Optimization*, SIAM, Philadelphia, PA, 1974.