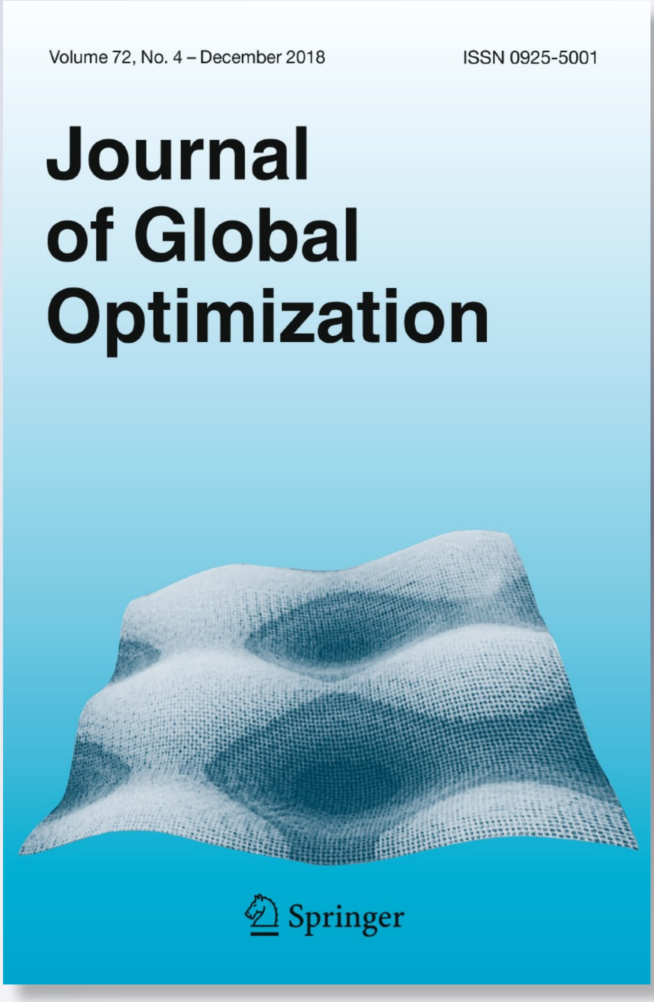
A DC programming approach for solving multicast network design problems via the Nesterov smoothing technique

W. Geremew, N. M. Nam, A. Semenov, V. Boginski & E. Pasiliao

Journal of Global Optimization
An International Journal Dealing with Theoretical and Computational Aspects of Seeking Global Optima and Their Applications in Science, Management and Engineering

ISSN 0925-5001
Volume 72
Number 4

J Glob Optim (2018) 72:705-729
DOI 10.1007/s10898-018-0671-9



Volume 72, No. 4 – December 2018 ISSN 0925-5001

Journal of Global Optimization

Springer

Springer

J Glob Optim (2018) 72:705–729
https://doi.org/10.1007/s10898-018-0671-9

A DC programming approach for solving multicast network design problems via the Nesterov smoothing technique

W. Geremew[1] · N. M. Nam[2] · A. Semenov[3] · V. Boginski[4,5] · E. Pasiliao[6]

Received: 28 September 2017 / Accepted: 1 June 2018 / Published online: 11 June 2018

Abstract This paper continues our recent effort in applying continuous optimization techniques to study optimal *multicast communication networks* modeled as bilevel hierarchical clustering problems. Given a finite number of nodes, we consider two different models of multicast networks by identifying a certain number of nodes as cluster centers, and at the same time, locating a particular node that serves as a total center so as to minimize the total transportation cost throughout the network. The fact that the cluster centers and the

Geremew's research was partly supported by the AFRL Mathematical Modeling and Optimization Institute. Nam's research was partly supported by the National Science Foundation under Grant DMS-1716057. Semenov's research was partly supported by the European Office of Aerospace Research and Development under Grant FA9550-17-1-0030.

✉ N. M. Nam
mau.nam.nguyen@pdx.edu

W. Geremew
geremeww@stockton.edu

A. Semenov
alexander.v.semenov@jyu.fi

V. Boginski
vladimir.boginski@ucf.edu

E. Pasiliao
elpasiliao@gmail.com

[1] School of General Studies, Stockton University, Galloway, NJ 08205, USA

[2] Fariborz Maseeh Department of Mathematics and Statistics, Portland State University, Portland, OR 97207, USA

[3] Faculty of Information Technology, University of Jyväskylä, P.O. Box 35, 40014 Jyväskylä, Finland

[4] Industrial Engineering & Management Systems, University of Central Florida, Orlando, FL 32816, USA

[5] Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611, USA

[6] Munitions Directorate, Air Force Research Laboratory, Building 13, Eglin AFB, FL 32542, USA

total center have to be among the given nodes makes these problems discrete optimization problems. Our approach is to reformulate the discrete problems as continuous ones and to apply Nesterov's smoothing approximation techniques on the Minkowski gauges that are used as distance measures. This approach enables us to propose two implementable DCA-based algorithms for solving the problems. Numerical results and practical applications are provided to illustrate our approach.

Keywords DC programming · Nesterov's smoothing techniques · Hierarchical clustering · Subgradient · Fenchel conjugate

Mathematics Subject Classification 49J52 · 49J53 · 90C31

1 Introduction

The complexity of modern networks such as communication networks, broadcasting networks, and distribution networks requires multilevel connectivity. For instance, many department stores usually get their merchandise delivered to them by a delivery company. For efficiency purposes, the delivery company wants to identify a certain number of locations to serve as distribution centers for the delivery of supplies to the stores. At the same time, the company also wants to identify a location as a main distribution center, known as the total center, from which the other distribution centers receive their supplies. This is a typical description of a bilevel multicast communication network, which can also be seen as a multifacility location problem or as a bilevel hierarchical clustering problem. Borrowing some language from the network optimization literature, the problem can be described mathematically as follows: Given m nodes $a^1, \ldots, a^m$ in $\mathbb{R}^n$, the objective is to choose k cluster centroids $a^{(1)}$, …, $a^{(k)}$ and a total center $a^{(k+1)}$ from the given nodes in such a way that the total *transportation cost* of the tree formed by connecting the cluster centers to the total center, and the remaining nodes to the nearest cluster centers is minimized. The fact that the centers and the total center have to be among the existing nodes makes the problem a discrete optimization problem, which can be shown to be NP-hard.

Many existing algorithms for solving bilevel hierarchical clustering problems are heuristic in nature, and do not optimize any well-defined objective function. The mathematical optimization approach for solving hierarchical clustering problems was initiated in the pioneering work from [6]. The authors introduced three models of hierarchical clustering based on the Euclidean norm and employed the derivative-free method developed in [5] to solve the problem in two dimensions. Replacing the Euclidean norm by the squared Euclidean norm, the authors in [3] used the DCA, a well-known algorithm for minimizing differences of convex functions introduced by Pham Dinh Tao (see [4,16]), to solve the problem in high dimensions. In fact, the DCA provides an effective tool for solving the classical clustering problem and its variants; see [1–3,6–8,11,12] and the references therein. In our recent work [12], we proposed a new method based on Nesterov's smoothing techniques and the DCA to cope with the original models of hierarchical clustering introduced in [6]. The idea of using Nesterov's smoothing techniques overcomes the drawback of the DCA stated in [3] as "the DCA is not appropriate for these models". Our current paper continues the effort initiated in [3,6] in which mathematical optimization techniques for solving optimization problems beyond convexity are used in multifacility location and clustering. This paper can be considered as the second part of our paper [12] as we propose two other bivelel hierarchical clustering models and investigate the possibility of considering problems with generalized distance generated by Minkowski gauges. We use two new ways to define the total centers compared with the existing methods from [12]. In [12], Model I defines the total center as the average of the k-centroids, while in the current paper it is directly determined by optimizing from the existing m nodes. In the second model of [12], the total center is determined by adding an additional variable in $\mathbb{R}^n$ to k cluster centers. On the other hand, in this paper we first model the total center itself as a cluster center, and then we determine its precise location by solving the corresponding optimization problem. In addition, using generalized distance measurements based on the Minkowski gauge allows us to solve clustering problems with different norms.

In this paper, we propose two implementable algorithms based on DC programming combined with Nesterov's smoothing techniques to solve the resulting constrained minimization problems for both models. It is important to note that the DCA can only guarantees the convergence to a critical point, so to achieve better results we often run the algorithms multiple times with different starting points for the two proposed algorithms.

The paper is organized as follows. In Sect. 2, we present the continuous optimization formulations of the two models using Minkowski gauges as distance measurements. In Sect. 3, we discuss some basic definitions and tools of optimization that are used throughout the paper. In Sects. 4 and 5, we develop the two algorithms for the two proposed multicast communication networks. In Sect. 6 we present our numerical experiments and results performed on artificial datasets as well as real datasets.

2 Problems formulation

In this section, we discuss two models of bilevel hierarchical clustering and provide the tools of optimization used throughout the paper. In order to reformulate the discrete optimization problem under consideration as a continuous optimization problem, we introduce k artificial centers which are not necessarily the existing nodes in designing the optimal multicast networks. Denote the k artificial cluster centers by $x^1, \ldots, x^k$ and the distance measurement between the artificial center x^ℓ, $\ell = 1, \ldots, k$, and the real node a^i, $i = 1, \ldots, m$, by a generalized distance $\sigma_F(x^\ell - a^i)$, where σ_F is the *support function* associated with a nonempty closed bounded convex set F containing the origin in its interior, i.e.,

$$\sigma_F(x) := \sup\{\langle x, y\rangle \mid y \in F\}.$$

Note that if F is the closed unit Euclidean ball in $\mathbb{R}^n$, then $\sigma_F(x)$ defines the Euclidean norm $\|x\|$ of $x \in \mathbb{R}^n$. In the case where F is the closed unit box of $\mathbb{R}^n$, i.e., $F := \{u = (u_1, \ldots, u_n) \in \mathbb{R}^n \mid -1 \le u_i \le 1 \text{ for } i = 1, \ldots, n\}$, then $\sigma_F(x)$ defines the ℓ^1-norm $\|x\|_1$ of $x \in \mathbb{R}^n$.

In the first model, the m nodes are clustered around the k artificial centers by trying to minimize the minimum sum of the distances from each node to the k cluster centers. A node with the smallest such sum will serve as the total center. The total connection cost of the tree that needs to be minimized is given by

$$\varphi_1(x^1, \ldots, x^k) := \sum_{i=1}^{m} \min_{\ell=1,\ldots,k} \sigma_F(x^\ell - a^i) \;+\; \min_{i=1,\ldots,m} \sum_{\ell=1}^{k} \sigma_F(x^\ell - a^i).$$

On the other hand, in the second model the m nodes are clustered around $k + 1$ artificial centers by trying to minimize the minimum sum of the distances from each artificial center to the remaining k centers. Such a center will eventually be named as the total center. In this case, the total connection cost of the tree that needs to be minimized is given by

$$\varphi_2(x^1,\ldots,x^{k+1}) := \sum_{i=1}^{m} \min_{\ell=1,\ldots,k+1} \sigma_F(x^\ell - a^i) \; + \; \min_{\ell=1,\ldots,k+1} \sum_{j=1}^{k+1} \sigma_F(x^\ell - x^j).$$

The main difference between Model I and Model II is the way in which the total center is selected. In addition, in Model II the total center also serves as a cluster center.

The algorithms we will develop are expected to solve the continuous optimization models in a reasonable amount of time and give us approximate solutions to the original discrete optimization models. Note that each node a^i is assigned to its closest center x^ℓ, but in both models the centers might not be real nodes. Therefore, for the continuous optimization model to solve (or approximate) the discrete model, we need to add a constraint that tries to minimize the difference between the artificial centers and the real centers, i.e.,

$$\phi_1(x^1,\ldots,x^k) := \sum_{\ell=1}^{k} \min_{i=1,\ldots,m} \sigma_F(x^\ell - a^i) = 0$$

and

$$\phi_2(x^1,\ldots,x^{k+1}) := \sum_{\ell=1}^{k+1} \min_{i=1,\ldots,m} \sigma_F(x^\ell - a^i) = 0.$$

Note that we use the generalized distance generated by σ_F in the constraints for convenience of presentation although it is possible to use different distances such as the Euclidean distance.

Model I was originally proposed in [6] where the authors used the *derivative-free discrete gradient method* established in [5] to solve the resulting optimization problem, but this method is not suitable for large-scale settings in high dimensions. It is also considered in [3] to solve a similar model where the *squared Euclidean distance* used as a similarity measure. Model II was considered in [8] without constraints, and the *hyperbolic smoothing technique* was used to solve the problem.

3 Basic definitions and tools of optimization

In this section, we present the main tools of optimization used to solve the proposed bilevel hierarchical clustering problems: the DCA and Nesterov's smoothing techniques.

We consider throughout the paper DC programming:

$$\text{minimize } f(x) := g(x) - h(x),\, x \in \mathbb{R}^n, \tag{3.1}$$

where $g\colon \mathbb{R}^n \to \mathbb{R}$ and $h\colon \mathbb{R}^n \to \mathbb{R}$ are convex functions. The function f in (3.1) is called a *DC function* and $g - h$ is called a *DC decomposition* of f.

Given a convex function $g\colon \mathbb{R}^n \to \mathbb{R}$, 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 $g^*\colon \mathbb{R}^n \to (-\infty, +\infty]$ 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., [9,10,15].

We will present below the DCA introduced by Tao and An [4,16] as applied to (3.1). Although the algorithm is used for nonconvex optimization problems, the convexity of the functions involved still plays a crucial role.

Algorithm 1 The DCA

```
Input: $x_0 \in \mathbb{R}^n$, $N \in \mathbb{N}$.
for $k = 1, \ldots, N$ 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 $h\colon \mathbb{R}^n \to \mathbb{R}$ 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 $\gamma-$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 (3.1) 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 [16, Theorem 3.7].

Theorem 3.1 *Consider the function f defined by (3.1) and the sequence $\{x_k\}$ generated by the Algorithm 1. Then 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.

In the next proposition, we present a direct consequence of Nesterov's smoothing techniques given in [13], where $d(x; \Omega)$ 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 3.2 *Given any $a \in \mathbb{R}^n$ and $\mu > 0$, a Nesterov smoothing approximation of $\varphi(x) := \sigma_F(x - a)$ has the representation*

$$\varphi_\mu(x) := \frac{1}{2\mu}\|x - a\|^2 - \frac{\mu}{2}\big[d(\frac{x - a}{\mu}; F)\big]^2.$$

Moreover, $\nabla\varphi_\mu(x) = P(\frac{x-a}{\mu}; F)$ and

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

where $\|F\| := \sup\{\|q\| \mid q \in F\}$.

4 Hierarchical clustering via continuous optimization techniques: Model I

In this section, we present an approach of using continuous optimization techniques for hierarchical clustering. As mentioned earlier, our main tools are the DCA and Nesterov's smoothing techniques. Recall that the first model under consideration is formulated as a constrained optimization problem:

$$\begin{aligned}&\text{minimize} \quad \sum_{i=1}^{m} \min_{\ell=1,\ldots,k} \sigma_F(x^\ell - a^i) + \min_{i=1,\ldots,m} \sum_{\ell=1}^{k} \sigma_F(x^\ell - a^i)\\ &\text{subject to} \quad \sum_{\ell=1}^{k} \min_{i=1,\ldots,m} \sigma_F(x^\ell - a^i) = 0, \ x^1, \ldots, x^k \in \mathbb{R}^n.\end{aligned}$$

After the centers $x^1, \ldots, x^k$ have been found, a total center is selected from the existing nodes as follows: For each $i = 1, \ldots, m$, we compute the sum $\sum_{\ell=1}^{k} \sigma_F(x^\ell - a^i)$. Then a total center c^* is a node a^i that yields the smallest sum, i.e.,

$$c^* := \text{ argmin} \left\{ \sum_{\ell=1}^{k} \sigma_F(x^\ell - a^i) \mid i = 1, \ldots, m \right\}.$$

Now we convert the constrained optimization problem under consideration to an unconstrained optimization problem using the penalty method with a penalty parameter $\lambda > 0$:

$$\begin{aligned}\text{minimize } &\sum_{i=1}^{m} \min_{\ell=1,\ldots,k} \sigma_F(x^\ell - a^i) + \min_{i=1,\ldots,m} \sum_{\ell=1}^{k} \sigma_F(x^\ell - a^i) + \lambda \sum_{\ell=1}^{k} \min_{i=1,\ldots,m} \sigma_F(x^\ell - a^i)\\ &x^1, \ldots, x^k \in \mathbb{R}^n.\end{aligned}$$

Proposition 4.1 *The objective function*

$$\begin{aligned}f(x^1, \ldots, x^k) := &\sum_{i=1}^{m} \min_{\ell=1,\ldots,k} \sigma_F(x^\ell - a^i) + \min_{i=1,\ldots,m} \sum_{\ell=1}^{k} \sigma_F(x^\ell - a^i)\\ &+ \lambda \sum_{\ell=1}^{k} \min_{i=1,\ldots,m} \sigma_F(x^\ell - a^i)\end{aligned}$$

for $x^1, \ldots, x^k \in \mathbb{R}^n$ *and* $\lambda > 0$ *can be written as a difference of convex functions.*

Proof First note that the minimum of m real numbers α_i for $i = 1, \ldots, m$ has the representation:

$$\min_{i=1,\ldots,m} \alpha_i = \sum_{i=1}^{m} \alpha_i - \max_{t=1,\ldots,m} \sum_{\substack{i=1\\ i \neq t}}^{m} \alpha_i.$$

Hence, we can represent $f(x^1, \ldots, x^k)$ as a function defined on $(\mathbb{R}^n)^k$ as follows:

$$\begin{aligned}f(x^1, \ldots, x^k) = (2+\lambda) &\sum_{i=1}^{m} \sum_{\ell=1}^{k} \sigma_F(x^\ell - a^i) - \sum_{i=1}^{m} \max_{t=1,\ldots,k} \sum_{\substack{\ell=1\\ \ell \neq t}}^{k} \sigma_F(x^\ell - a^i)\\ &- \lambda \sum_{\ell=1}^{k} \max_{t=1,\ldots,m} \sum_{\substack{i=1\\ i \neq t}}^{m} \sigma_F(x^\ell - a^i) - \max_{t=1,\ldots,m} \sum_{\substack{i=1\\ i \neq t}}^{m} \sum_{\ell=1}^{k} \sigma_F(x^\ell - a^i).\end{aligned}$$

This shows that f has a DC representation $f = g_0 - h_0$, where

$$g_0(x^1, \ldots, x^k) := (2+\lambda)\sum_{i=1}^{m}\sum_{\ell=1}^{k}\sigma_F(x^\ell - a^i) \tag{4.1}$$

and

$$\begin{aligned} h_0(x^1, \ldots, x^k) := &\sum_{i=1}^{m}\max_{t=1,\ldots,k}\sum_{\substack{\ell=1\\ \ell\neq t}}^{k}\sigma_F(x^\ell - a^i) + \lambda\sum_{\ell=1}^{k}\max_{t=1,\ldots,m}\sum_{\substack{i=1\\ i\neq t}}^{m}\sigma_F(x^\ell - a^i)\\ &+ \max_{t=1,\ldots,m}\sum_{\substack{i=1\\ i\neq t}}^{m}\sum_{\ell=1}^{k}\sigma_F(x^\ell - a^i)\end{aligned}$$

are convex functions defined on $(\mathbb{R}^n)^k$. □

Based on Proposition 3.2, we obtain a Nesterov's approximation of the generalized distance function $\varphi(x) := \sigma_F(x-a)$ for $x, a \in \mathbb{R}^n$ as follows

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

As a result, the function g_0 defined in (4.1) has a smooth approximation given by

$$\begin{aligned} g_{0\mu}(x^1, \ldots, x^k) := &\frac{(2+\lambda)\mu}{2}\sum_{i=1}^{m}\sum_{\ell=1}^{k}\left\|\frac{x^\ell - a^i}{\mu}\right\|^2\\ &- \frac{(2+\lambda)\mu}{2}\sum_{i=1}^{m}\sum_{\ell=1}^{k}\left[d\left(\frac{x^\ell - a^i}{\mu}; F\right)\right]^2.\end{aligned}$$

Thus, the function f has the following DC approximation convenient for applying the DCA:

$$\begin{aligned} f_\mu(x^1, \ldots, x^k) := &\frac{(2+\lambda)\mu}{2}\sum_{i=1}^{m}\sum_{\ell=1}^{k}\left\|\frac{x^\ell - a^i}{\mu}\right\|^2 - \frac{(2+\lambda)\mu}{2}\sum_{i=1}^{m}\sum_{\ell=1}^{k}\left[d\left(\frac{x^\ell - a^i}{\mu}; F\right)\right]^2\\ &- \sum_{i=1}^{m}\max_{t=1,\ldots,k}\sum_{\substack{\ell=1\\ \ell\neq t}}^{k}\sigma_F(x^\ell - a^i) - \lambda\sum_{\ell=1}^{k}\max_{t=1,\ldots,m}\sum_{\substack{i=1\\ i\neq t}}^{m}\sigma_F(x^\ell - a^i)\\ &- \max_{t=1,\ldots,m}\sum_{\substack{i=1\\ i\neq t}}^{m}\sum_{\ell=1}^{k}\sigma_F(x^\ell - a^i).\end{aligned}$$

Instead of minimizing the function f, we minimize its DC approximation

$$f_\mu(x^1, \ldots, x^k) = g_\mu(x^1, \ldots, x^k) - h_\mu(x^1, \ldots, x^k),\ \ x^1, \ldots, x^k \in \mathbb{R}^n.$$

In this formulation, g_μ and h_μ are convex functions given by

$$\begin{aligned} g_\mu(x^1,\ldots,x^k) &:= \frac{2+\lambda}{2\mu}\sum_{i=1}^{m}\sum_{\ell=1}^{k}\|x^\ell - a^i\|^2,\\ h_\mu(x^1,\ldots,x^k) &:= h_{1\mu}(x^1,\ldots,x^k) + h_2(x^1,\ldots,x^k) + h_3(x^1,\ldots,x^k)\\ &\quad + h_4(x^1,\ldots,x^k), \end{aligned}$$

where

$$\begin{aligned} h_{1\mu}(x^1,\ldots,x^k) &:= \frac{(2+\lambda)\mu}{2}\sum_{i=1}^{m}\sum_{\ell=1}^{k}\left[d\left(\frac{x^\ell - a^i}{\mu};F\right)\right]^2,\ h_2(x^1,\ldots,x^k)\\ &:= \sum_{i=1}^{m}\max_{t=1,\ldots,k}\sum_{\substack{\ell=1\\ \ell\neq t}}^{k}\sigma_F(x^\ell - a^i),\\ h_3(x^1,\ldots,x^k) &:= \lambda\sum_{\ell=1}^{k}\max_{t=1,\ldots,m}\sum_{\substack{i=1\\ i\neq t}}^{m}\sigma_F(x^\ell - a^i),\ h_4(x^1,\ldots,x^k)\\ &:= \max_{t=1,\ldots,m}\sum_{\substack{i=1\\ i\neq t}}^{m}\sum_{\ell=1}^{k}\sigma_F(x^\ell - a^i). \end{aligned}$$

The proposition below is a direct consequence of Proposition 3.2.

Proposition 4.2 *Given any $\lambda > 0$ and $\mu > 0$, the functions f and f_μ satisfy*

$$f_\mu(x^1,\ldots,x^k) \le f(x^1,\ldots,x^k) \le f_\mu(x^1,\ldots,x^k) + mk\left(1+\frac{\lambda}{2}\right)\mu\|F\|^2.$$

for all $x^1,\ldots,x^k \in \mathbb{R}^n$.

In what follows we will prove that each of the functions f and f_μ admits an absolute minimum in $(\mathbb{R}^n)^k$.

Theorem 4.3 *Given any $\lambda > 0$ and $\mu > 0$, each of the functions f and f_μ has an absolute minimum in $(\mathbb{R}^n)^k$.*

Proof Let us show that for any $\gamma \in \mathbb{R}$, the sublevel set

$$\mathcal{L}_\gamma := \{(x^1,\ldots,x^k) \mid f(x^1,\ldots,x^k) \le \gamma\}$$

is bounded in $(\mathbb{R}^n)^k$. Since $0 \in \operatorname{int}(F)$, there exists $r > 0$ such that $\mathbb{B}(0;r) \subset F$. Consequently,

$$r\|x\| = \sup\{\langle x,u\rangle \mid u \in \mathbb{B}(0;r)\} \le \sup\{\langle x,u\rangle \mid u \in F\} = \sigma_F(x)\ \text{ for all } x \in \mathbb{R}^n.$$

From the definition of the function f, we have

$$\begin{aligned}
&\{(x^1,\ldots,x^k)\in(\mathbb{R}^n)^k \mid f(x^1,\ldots,x^k)\le\gamma\}\subset\Big\{(x^1,\ldots,x^k)\\
&\in(\mathbb{R}^n)^k \mid \min_{i=1,\ldots,m}\sum_{\ell=1}^{k}\sigma_F(x^\ell-a^i)\le\gamma\Big\}\\
&\subset\Big\{(x^1,\ldots,x^k)\in(\mathbb{R}^n)^k \mid \min_{i=1,\ldots,m}\sum_{\ell=1}^{k}\|x^\ell-a^i\|\le\frac{\gamma}{r}\Big\}\\
&\subset\bigcup_{i=1}^{m}\Big\{(x^1,\ldots,x^k)\mid\varphi_i(x^1,\ldots,x^k)\le\frac{\gamma}{r}\Big\},
\end{aligned}$$

where $\varphi_i(x^1,\ldots,x^k):=\sum_{\ell=1}^{k}\|x^\ell-a^i\|$. Observe that for each $i=1,\ldots,m$, one has the inclusion

$$\Big\{(x^1,\ldots,x^k)\mid\varphi_i(x^1,\ldots,x^k)\le\frac{\gamma}{r}\Big\}\subset\{(x^1,\ldots,x^k)\mid\sum_{\ell=1}^{k}\|x^\ell\|\le\frac{\gamma}{r}+k\|a^i\|\}.$$

Thus, $\mathcal{L}_\gamma$ is a bounded set as it is contained in the union of a finite number of bounded sets in $(\mathbb{R}^n)^k$. As f is a continuous function, it has an absolute minimum in $(R^n)^k$.

Let $\gamma_\mu:=mk\left(1+\frac{\lambda}{2}\right)\mu\|F\|^2$. It follows from Proposition 4.2 that for any $\gamma\in\mathbb{R}$,

$$\begin{aligned}
&\Big\{(x^1,\ldots,x^k)\in(\mathbb{R}^n)^k\mid f_\mu(x^1,\ldots,x^k)\le\gamma\Big\}\\
&\quad\subset\Big\{(x^1,\ldots,x^k)\in(\mathbb{R}^n)^k\mid f(x^1,\ldots,x^k)\le\gamma_\mu+\gamma\Big\}.
\end{aligned}$$

It follows that the sublevel set $\{(x^1,\ldots,x^k)\in(\mathbb{R}^n)^k\mid f_\mu(x^1,\ldots,x^k)\le\gamma\}$ is also bounded, and hence f_μ has an absolute minimum in $(\mathbb{R}^n)^k$. □

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

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

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

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

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

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

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

$$g_\mu(\mathbf{X}) \;=\; \frac{2+\lambda}{2\mu}\sum_{i=1}^{m}\sum_{\ell=1}^{k}\|x^\ell - a^i\|^2.$$

Using the Frobenius norm, the function g_μ can be written as

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

where $\mathbf{E}$ is a $k \times m$ matrix whose entries are all ones. Hence, g_μ is differentiable and its gradient is given by

$$\nabla g_\mu(\mathbf{X}) = \frac{2+\lambda}{\mu}\left[m\mathbf{X} - \mathbf{EA}\right].$$

Our goal now is to find $\mathbf{X} \in \partial g^*(\mathbf{Y})$, which can be accomplished by employing the relation

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

This can equivalently be written as $\frac{2+\lambda}{\mu}\left[m\mathbf{X} - \mathbf{EA}\right] = \mathbf{Y}$, and we solve for $\mathbf{X}$ as follows:

$$\begin{aligned} (2+\lambda)\left[m\mathbf{X} - \mathbf{EA}\right] &= \mu\mathbf{Y} \\ (2+\lambda)\mathbf{X} &= (2+\lambda)\mathbf{EA} + \mu\mathbf{Y} \\ \mathbf{X} &= \frac{(2+\lambda)\mathbf{EA} + \mu\mathbf{Y}}{(2+\lambda)m} \end{aligned}$$

Next, we will demonstrate in more detail the techniques we used to compute a subgradient for the convex function

$$h_\mu = h_{1\mu} + \sum_{j=2}^{4} h_j.$$

Since each function in this sum is convex, we will compute a subgradient of h_μ applying the subdifferential sum rule (see, e.g., [10, Corollary 2.46]) and maximum rule (see, e.g., [10, Proposition 2.54]) well known in convex analysis. We will begin our demonstration with $h_{1\mu}$ given by

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

From its representation one can see that $h_{1\mu}$ is differentiable. Thus, its gradient at $\mathbf{X}$ can be computed by computing the *partial derivatives* with respect to $x^1, \dots, x^k$, i.e.,

$$\frac{\partial h_{1\mu}}{\partial x^\ell}(\mathbf{X}) = (2+\lambda)\sum_{i=1}^{m}\left[\frac{x^\ell - a^i}{\mu} - P\left(\frac{x^\ell - a^i}{\mu}; F\right)\right] \text{ for } \ell = 1, \dots, k. \tag{4.2}$$

Hence, $\nabla h_{1\mu}(\mathbf{X}))$ is a $k \times n$ matrix $\mathbf{H}_1$ whose ℓ^{th} row is $\frac{\partial h_{1\mu}}{\partial x^\ell}(\mathbf{X}))$.

Note that the convex functions h_j for $j = 2, 3, 4$ are not differentiable in general. However, we can compute a subgradient for each function at $\mathbf{X}$ by applying the subdifferential sum rule and maximum rule for convex functions. The following is an illustration of how one can compute subgradients of such functions using h_2 as an example. For $t = 1, \dots, k$ and $i = 1, \dots, m$, define

$$\gamma_{ti}(\mathbf{X}) := \sum_{\ell=1,\ell\neq t}^{k} \sigma_F(x^\ell - a^i) = \sum_{\ell=1}^{k}\sigma_F(x^\ell - a^i) - \sigma_F(x^t - a^i) \text{ and } \gamma_i(\mathbf{X})$$
$$:= \max_{t=1,\dots,k} \gamma_{ti}(\mathbf{X}).$$

Thus, h_2 can be represented as the sum of m convex functions as follows:

$$h_2(\mathbf{X}) = \sum_{i=1}^{m}\max_{t=1,\dots,k}\sum_{\ell=1,\ell\neq t}^{k}\sigma_F(x^\ell - a^i) = \sum_{i=1}^{m}\gamma_i(\mathbf{X}).$$

Note that γ_i is the maximum of k convex functions γ_{ti} for $t = 1, \dots, k$. Based on the subdifferential maximum rule, for each $i = 1, \dots, m$, we will find a $k \times n$ matrix $\mathbf{H}_{2i} \in \partial\gamma_i(\mathbf{X})$. Then, by the subdifferential sum rule $\mathbf{H}_2 := \sum_{i=1}^{m} \mathbf{H}_{2i}$ is a subgradient of h_2 at $\mathbf{X}$. To accomplish this goal, we first choose an index $t^* \in \{1, \dots, k\}$ such that $\gamma_i(\mathbf{X}) = \gamma_{t^*i}(\mathbf{X}) := \sum_{\ell=1,\ell\neq t^*}^{k} \sigma_F(x^\ell - a^i)$. The ℓth row w^i_ℓ of the matrix $\mathbf{H}_{2i}$ for $\ell \neq t^*$ can be computed as described in Proposition 4.4 below, which follows from [10, Theorem 2.93]. The t^* row of the matrix $\mathbf{H}_{2i}$ is set to zero, as γ_{it^*} is independent of x^{t^*}. The procedures for computing a subgradient for h_3 and h_4 are very similar to the procedure we have illustrated.

Proposition 4.4 *Given $a \in \mathbb{R}^n$, the function $\varphi(x) := \sigma_F(x - a)$ is convex with its subdifferential at $\bar{x} \in \mathbb{R}^n$ given by*

$$\partial\varphi(\bar{x}) = co\, F(\bar{x}),$$

where $F(\bar{x}) := \{q \in F \mid \langle \bar{x}, q\rangle = \sigma_F(\bar{x})\}$.

In particular, if F is the Euclidean closed unit ball in $\mathbb{R}^n$, then

$$\partial\varphi(\bar{x}) = \begin{cases} \frac{\bar{x}-a}{\|\bar{x}-a\|} & \text{if } \bar{x} \neq a, \\ \mathbb{B} & \text{if } \bar{x} = a. \end{cases}$$

At this point, we have demonstrated all the necessary steps in calculating the gradients and subgradients needed for our first DCA-based algorithm for solving the bilevel hierarchical clustering problem formulated in Model I.

Example 4.5 (ℓ^2-clustering with Algorithm 2). In this example, we illustrate our method to study the problem of ℓ^2-clustering. The key point in Algorithm 2 is the computation of $\mathbf{Y} \in \partial h_\mu(\mathbf{X})$ for the case where F is the Euclidean closed unit ball $\mathbb{B}$ in $\mathbb{R}^n$. By the subdifferential sum rule,

$$h_\mu(\mathbf{X}) = \nabla h_{1\mu}(\mathbf{X}) + \partial h_2(\mathbf{X}) + \partial h_3(\mathbf{X}) + \partial h_4(\mathbf{X}).$$

Algorithm 2 Model I

```
Input: $\mathbf{A}, \mathbf{X}_0, \lambda_0, \mu_0, \sigma_1, \sigma_2, \varepsilon, N \in \mathbb{N}$.
while stopping criteria $(\lambda, \mu, \varepsilon)$ = false do
    for $k = 1, \ldots, N$ do
        Find $\mathbf{Y}_k \in \partial h_\mu(\mathbf{X}_{k-1})$
        $\mathbf{X}_k = \frac{(2+\lambda)\mathbf{EA}+\mu\mathbf{Y}_k}{(2+\lambda)m}$
    end for
    update $\lambda$ and $\mu$
end while
Output: $x_N$.
```

Define

$$u_{\ell i} := \begin{cases} \frac{x^\ell - a^i}{\|x^\ell - a^i\|} & \text{if } x^\ell \neq a^i, \\ 0 & \text{otherwise.} \end{cases}$$

Now, we illustrate the way to find the gradient of h_1 and a subgradient of h_i for $i = 2, 3, 4$ at $\mathbf{X}$.

The gradient of h_1: The gradient $\mathbf{Y}_1 := \nabla h_1(\mathbf{X})$ is the $k \times n$ matrix whose ℓth row is $\frac{\partial h_{1\mu}}{\partial x^\ell}(\mathbf{X})$ given in (4.2). Note that in this case, the Euclidean projection $P(z; F)$ from $z \in \mathbb{R}^n$ to F is given by

$$P(z; F) := \begin{cases} \frac{z}{\|z\|} & \text{if } \|z\| > 1, \\ z & \text{otherwise.} \end{cases}$$

A subgradient of h_2: In this case,

$$h_2(\mathbf{X}) = \sum_{i=1}^{m} \max_{t=1,\ldots,k} \sum_{\substack{\ell=1 \\ \ell \neq t}}^{k} \|x^\ell - a^i\| = \sum_{i=1}^{m} \max_{t=1,\ldots,k} \left(\sum_{\ell=1}^{k} \|x^\ell - a^i\| - \|x^t - a^i\| \right).$$

For each $i = 1, \ldots, m$, choose an index $t(i)$ such that

$$\max_{t=1,\ldots,k} \left(\sum_{\ell=1}^{k} \|x^\ell - a^i\| - \|x^t - a^i\| \right) = \sum_{\ell=1}^{k} \|x^\ell - a^i\| - \|x^{t(i)} - a^i\|.$$

Let us now form a $k \times mn$ *block matrix* $\mathbf{U} = (u_{\ell i})$, where $u_{\ell i}$ is considered as a row vector. We also use U^i to denote the ith block column of the matrix $\mathbf{U}$. Equivalently, U_i is the $k \times n$ matrix formed by placing the row vectors $u_{\ell i}$ in its ℓth row for $\ell = 1, \ldots, k$. Then a subgradient of h_2 at $\mathbf{X}$ is given by

$$\mathbf{Y}_2 := \sum_{i=1}^{m} \left(U^i - e_{t(i)} u_{t(i)i} \right),$$

where $e_{t(i)}$ is the column vector of k components with 1 at the $t(i)$th position and 0 at other positions.

A subgradient of h_3: In this case,

$$h_3(\mathbf{X}) = \lambda \sum_{\ell=1}^{k} \max_{t=1,\ldots,m} \sum_{\substack{i=1 \\ i \neq t}}^{m} \|x^\ell - a^i\| = \lambda \sum_{\ell=1}^{k} \max_{t=1,\ldots,m} \left(\sum_{i=1}^{m} \|x^\ell - a^i\| - \|x^\ell - a^t\| \right).$$

For each $\ell = 1, \ldots, k$, we choose an index $t(\ell)$ such that

$$\max_{t=1,\ldots,m} \left(\sum_{i=1}^{m} \|x^\ell - a^i\| - \|x^\ell - a^t\| \right) = \sum_{i=1}^{m} \|x^\ell - a^i\| - \|x^\ell - a^{t(\ell)}\|.$$

Let $\mathbf{V}$ be the $k \times n$ matrix whose ℓth row is $\sum_{i=1}^{m} u_{\ell i} - u_{\ell t(\ell)}$. Then a subgradient of h_3 at X is given by

$$\mathbf{Y}_3 := \lambda \mathbf{V}.$$

A subgradient of h_4: In this case,

$$\begin{aligned} h_4(\mathbf{X}) &= \max_{t=1,\ldots,m} \sum_{\substack{i=1 \\ i \neq t}}^{m} \sum_{\ell=1}^{k} \|x^\ell - a^i\| \\ &= \max_{t=1,\ldots,m} \left(\sum_{i=1}^{m} \sum_{\ell=1}^{k} \|x^\ell - a^i\| - \sum_{\ell=1}^{k} \|x^\ell - a^t\| \right). \end{aligned}$$

Again, we choose an index t such that

$$\max_{t=1,\ldots,m} \left(\sum_{i=1}^{m} \sum_{\ell=1}^{k} \|x^\ell - a^i\| - \sum_{\ell=1}^{k} \|x^\ell - a^t\| \right) = \sum_{i=1}^{m} \sum_{\ell=1}^{k} \|x^\ell - a^i\| - \sum_{\ell=1}^{k} \|x^\ell - a^t\|.$$

Let $\mathbf{Z}$ be the $k \times n$ matrix whose ℓth row is $\sum_{i=1}^{m} u_{\ell i}$. Then a subgradient of h_4 is given by

$$\mathbf{Y}_4 := Z - Z_t,$$

where Z_t is the $k \times n$ matrix whose ℓth row is $u_{\ell t}$.

Example 4.6 (ℓ^1-clustering with Algorithm 2). In this example, we illustrate our method to study the problem of ℓ^1-clustering. We will find a subgradient $\mathbf{Y} \in \partial h_\mu(\mathbf{X})$ for the case where F is the *closed unit box* in $\mathbb{R}^n$ given by

$$F := \{(u_1, \ldots, u_n) \in \mathbb{R}^n \mid -1 \leq u_i \leq 1 \ \text{ for } i = 1, \ldots, n\}.$$

For $t \in \mathbb{R}$, define

$$\text{sign}(t) := \begin{cases} 1 & t > 0, \\ 0 & t = 0, \\ -1 & t < 0. \end{cases}$$

Then we define $\text{sign}(x) := (\text{sign}(x_1), \ldots, \text{sign}(x_n))$ for $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$. Note that for the function $p(x) := \|x\|_1$, a subgradient of p at $x \in \mathbb{R}^n$ is simply $\text{sign}(x)$. Now, we illustrate the way to find the gradient of h_1 and a subgradient of h_i for $i = 2, 3, 4$ at $\mathbf{X}$.
The gradient of h_1: Similar to Example 4.5, the gradient of $\mathbf{Y}_1 := \nabla h_1(\mathbf{X})$ is the $k \times n$ matrix whose ℓth row is $\frac{\partial h_{1\mu}}{\partial x^\ell}(\mathbf{X})$ given in (4.2). Note that in this case, the Euclidean projection $P(z; F)$ from $z \in \mathbb{R}^n$ to F is given by

$$P(z; F) := \max(-e, \min(z, e)) \ \text{ componentwise},$$

where $e \in \mathbb{R}^n$ is the vector consisting of 1 in each component.

A subgradient of h_2: In this case,

$$h_2(\mathbf{X}) = \sum_{i=1}^{m} \max_{r=1,\ldots,k} \sum_{\substack{\ell=1\\ \ell\neq r}}^{k} \|x^\ell - a^i\|_1 = \sum_{i=1}^{m} \max_{r=1,\ldots,k} \left(\sum_{\ell=1}^{k} \|x^\ell - a^i\|_1 - \|x^r - a^i\|_1 \right).$$

For each $i = 1, \ldots, m$, choose an index $r(i)$ such that

$$\max_{r=1,\ldots,k} \left(\sum_{\ell=1}^{k} \|x^\ell - a^i\|_1 - \|x^r - a^i\|_1 \right) = \sum_{\ell=1}^{k} \|x^\ell - a^i\|_1 - \|x^{r(i)} - a^i\|_1.$$

Now we form the $k \times mn$ *signed block matrix* $S = (s_{\ell i})$ given by $s_{\ell i} = \operatorname{sign}(x^\ell - a^i)$ as a row vector. We also use S^i to denote the ith column block matrix of the signed matrix S. Then a subgradient of h_2 at $\mathbf{X}$ is given by

$$\mathbf{Y}_2 := \sum_{i=1}^{m} \left(S^i - e_{r(i)} s_{r(i)i} \right),$$

where $e_{r(i)}$ is the column vector of k components with 1 at the $r(i)$th position and 0 at other positions.

A subgradient of h_3: In this case,

$$h_3(\mathbf{X}) = \lambda \sum_{\ell=1}^{k} \max_{t=1,\ldots,m} \sum_{\substack{i=1\\ i\neq t}}^{m} \sigma_F(x^\ell - a^i) = \lambda \sum_{\ell=1}^{k} \max_{t=1,\ldots,m} \left(\sum_{i=1}^{m} \|x^\ell - a^i\|_1 - \|x^\ell - a^t\|_1 \right).$$

For each $\ell = 1, \ldots, k$, we choose an index $t(\ell)$ such that

$$\max_{t=1,\ldots,m} \left(\sum_{i=1}^{m} \|x^\ell - a^i\|_1 - \|x^\ell - a^t\|_1 \right) = \sum_{i=1}^{m} \|x^\ell - a^i\|_1 - \|x^\ell - a^{t(\ell)}\|_1.$$

Let $\mathbf{V}$ be the $k \times n$ matrix whose ℓth row is $\sum_{i=1}^{m} s_{\ell i} - s_{\ell t(\ell)}$. Then a subgradient of h_3 at $\mathbf{X}$ is given by

$$\mathbf{Y}_3 := \lambda \mathbf{V}.$$

A subgradient of h_4: In this case,

$$\begin{aligned} h_4(\mathbf{X}) &= \max_{t=1,\ldots,m} \sum_{\substack{i=1\\ i\neq t}}^{m} \sum_{\ell=1}^{k} \|x^\ell - a^i\|_1 \\ &= \max_{t=1,\ldots,m} \left(\sum_{i=1}^{m} \sum_{\ell=1}^{k} \|x^\ell - a^i\|_1 - \sum_{\ell=1}^{k} \|x^\ell - a^t\|_1 \right). \end{aligned}$$

Again, we choose an index t such that

$$\max_{t=1,\ldots,m} \left(\sum_{i=1}^{m} \sum_{\ell=1}^{k} \|x^\ell - a^i\|_1 - \sum_{\ell=1}^{k} \|x^\ell - a^t\|_1 \right) = \sum_{i=1}^{m} \sum_{\ell=1}^{k} \|x^\ell - a^i\|_1 - \sum_{\ell=1}^{k} \|x^\ell - a^t\|_1.$$

Let T be the $k \times n$ matrix whose ℓth row is $\sum_{i=1}^{m} s_{\ell i}$. Then a subgradient of h_4 is given by

$$\mathbf{Y}_4 := T - T_t,$$

where T_t is the $k \times n$ matrix whose ℓth row is $s_{\ell t}$.

5 Hierarchical clustering via continuous optimization techniques: Model II

In this section, we focus on developing nonconvex optimization techniques based on the DCA and Nesterov's smoothing techniques for the second model. Similar to Model I, we will solve the following constrained optimization problem:

$$\begin{aligned}\text{minimize} \quad & \sum_{i=1}^{m} \min_{\ell=1,\ldots,k+1} \sigma_F(x^\ell - a^i) + \min_{\ell=1,\ldots,k+1} \sum_{j=1}^{k+1} \sigma_F(x^\ell - x^j)\\ \text{subject to} \quad & \sum_{\ell=1}^{k+1} \min_{i=1,\ldots,m} \sigma_F(x^\ell - a^i) = 0,\ x^1, \ldots, x^{k+1} \in \mathbb{R}^n.\end{aligned}$$

The total center is determined by

$$c^* := \ \operatorname{argmin} \left\{ \sum_{j=1}^{k+1} \sigma_F(x^\ell - x^j) \mid \ell = 1, \ldots, k+1 \right\}.$$

This constrained optimization problem can be solved by the following unconstrained optimization problem by the penalty method with a penalty parameter $\lambda > 0$:

$$\begin{aligned}\text{minimize} & \sum_{i=1}^{m} \min_{\ell=1,\ldots,k+1} \sigma_F(x^\ell - a^i) + \min_{\ell=1,\ldots,k+1} \sum_{j=1}^{k+1} \sigma_F(x^\ell - x^j)\\ & + \lambda \sum_{\ell=1}^{k+1} \min_{i=1,\ldots,m} \sigma_F(x^\ell - a^i) x^1, \ldots, x^{k+1} \in \mathbb{R}^n.\end{aligned}$$

With Nesterov's smoothing techniques, the objective function has the following approximation that is convenient for implementing the DCA:

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

As in the previous section, we use a variable matrix $\mathbf{X}$ of size $(k+1) \times n$ to store the row vector x^ℓ in its ℓth row for $\ell = 1, \ldots, k+1$. Now we solve the following DC programming:

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

where g_μ and h_μ are convex functions by

$$g_\mu(\mathbf{X}) := g_{1\mu}(\mathbf{X}) + g_{2\mu}(\mathbf{X}) \tag{5.1}$$

and

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

where their respective components are defined as follows:

$$g_{1\mu}(\mathbf{X}) := \frac{(1+\lambda)\mu}{2}\sum_{i=1}^{m}\sum_{\ell=1}^{k+1}\left\|\frac{x^\ell - a^i}{\mu}\right\|^2,\ g_{2\mu}(\mathbf{X}) = \frac{\mu}{2}\sum_{\ell=1}^{k+1}\sum_{j=1}^{k+1}\left\|\frac{x^\ell - x^j}{\mu}\right\|^2$$

and

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

$$h_{2\mu}(\mathbf{X}) := \frac{\mu}{2}\sum_{\ell=1}^{k+1}\sum_{j=1}^{k+1}\left[d\left(\frac{x^\ell - x^j}{\mu}; F\right)\right]^2$$

$$h_3(\mathbf{X}) := \sum_{i=1}^{m}\max_{t=1,\ldots,k+1}\sum_{\substack{\ell=1\\ \ell\neq t}}^{k+1}\sigma_F(x^\ell - a^i),\ h_4(\mathbf{X}) := \lambda\sum_{\ell=1}^{k+1}\max_{t=1,\ldots,m}\sum_{\substack{i=1\\ i\neq t}}^{m}\sigma_F(x^\ell - a^i),$$

$$h_5(\mathbf{X}) := \max_{t=1,\ldots,k+1}\sum_{\substack{\ell=1\\ \ell\neq t}}^{k+1}\sum_{j=1}^{k+1}\sigma_F(x^\ell - x^j).$$

Lemma 5.1 *Let* $\mathbf{E}$ *be square matrix with size* $(k+1)$ *whose entries are all ones and let* $\mathbb{I}$ *be the identity matrix of size* $(k+1)$.
(i) Given any real numbers a *and* b *with* $a \neq 0$ *and* $a \neq -(k+1)b$*, the matrix* $\mathbf{M} := a\mathbb{I} + b\mathbf{E}$ *is invertible with*

$$\mathbf{M}^{-1} = x\mathbb{I} + y\mathbf{E},$$

where $x = \dfrac{1}{a}$ *and* $y = -\dfrac{b}{a[a+b(k+1)]}$.
(ii) Let $\widetilde{\mathbf{E}} := (k+1)\mathbb{I} - \mathbf{E}$*. Given any real numbers* c *and* d *with* $c \neq 0$ *and* $c \neq -d(k+1)$*, the matrix* $\mathbf{N} := c\mathbb{I} + d\widetilde{\mathbf{E}}$ *is invertible with*

$$\mathbf{N}^{-1} = \alpha\mathbb{I} + \beta\mathbf{E},$$

where $\alpha = \frac{1}{c+d(k+1)}$ *and* $\beta = \dfrac{d}{c[c+d(k+1)]}$.

Proof (i) Observe that

$$\begin{aligned}(a\mathbb{I} + b\mathbf{E})(x\mathbb{I} + y\mathbf{E}) &= ax\mathbb{I} + (bx+ay)\mathbf{E} + by\mathbf{E}^2\\ &= ax\mathbb{I} + (bx+ay)\mathbf{E} + by(k+1)\mathbf{E}.\end{aligned}$$

Thus, $(a\mathbb{I} + b\mathbf{E})(x\mathbb{I} + y\mathbf{E}) = \mathbb{I}$ if and only if

$$ax = 1 \ \text{ and } bx + [a + b(k+1)]y = 0.$$

Equivalently, $x = \dfrac{1}{a}$ and $y = -\dfrac{b}{a[a+b(k+1)]}$.
(ii) We have

$$N = c\mathbb{I} + d\widetilde{\mathbf{E}} = [c + d(k+1)\mathbb{I}] - d\mathbf{E}.$$

It remains to apply the result from (i). □

The proposition below provides a formula for computing ∇g_μ^* required for applying the DCA.

Proposition 5.2 *Given any* $\lambda > 0$ *and* $\mu > 0$, *the Fenchel conjugate* g_μ^* *of the function* g_μ *defined in* (5.1) *is continuously differentiable with*

$$\nabla g_\mu^*(\mathbf{Y}) = (\alpha\mathbb{I} + \beta\mathbf{E})\Big((1+\lambda)\mathbf{E}\mathbf{A} + \mu\mathbf{Y}\Big) \text{ for } \mathbf{Y} \in \mathbb{R}^{k\times n},$$

where $\mathbf{E}$ *is defined in Lemma* 5.1 *and*

$$\alpha := \frac{1}{m(\lambda+1) + 2(k+1)} \text{ and } \beta := \frac{2}{m(\lambda+1)[m(\lambda+1) + 2(k+1)]}. \tag{5.2}$$

Proof We have

$$\begin{aligned} \nabla g_{1\mu}(\mathbf{X}) &= \frac{1+\lambda}{\mu}[m\mathbf{X} - \mathbf{E}\mathbf{A}], \\ \nabla g_{2\mu}(\mathbf{X}) &= \frac{2}{\mu}[(k+1)\mathbb{I} - \mathbf{E}]\,\mathbf{X}. \end{aligned}$$

Recall that $\mathbf{X} \in \partial g_\mu^*(\mathbf{Y})$ if and only if $Y = \nabla g_\mu(\mathbf{X})$. The equation $\nabla g_\mu(\mathbf{X}) = \mathbf{Y}$ can be written as

$$\begin{aligned} &\frac{1+\lambda}{\mu}[m\mathbf{X} - \mathbf{E}\mathbf{A}] + \frac{2}{\mu}\widetilde{\mathbf{E}}\mathbf{X} = \mathbf{Y} \\ &(1+\lambda)[m\mathbf{X} - \mathbf{E}\mathbf{A}] + 2\widetilde{\mathbf{E}}\mathbf{X} = \mu\mathbf{Y} \\ &\big(m(1+\lambda)\mathbb{I} + 2\widetilde{\mathbf{E}}\big)\mathbf{X} = (1+\lambda)\mathbf{E}\mathbf{A} + \mu\mathbf{Y}. \end{aligned}$$

Solving this equation using Lemma 5.1(ii) yields

$$\mathbf{X} = (\alpha\mathbb{I} + \beta\mathbf{E})\Big((1+\lambda)\mathbf{E}\mathbf{A} + \mu\mathbf{Y}\Big), \tag{5.3}$$

where α and β are given in (5.2). It follows that $\partial g_\mu^*(\mathbf{Y})$ is a singleton for every $\mathbf{Y} \in \mathbb{R}^{k\times n}$, and so g_μ^* is continuously differentiable and $\nabla g_\mu^*(\mathbf{Y})$ is given by the expression on the right-hand side of (5.3); see [10, Theorem 3.3]. □

To implement the DCA, it remains to find a subgradient of h_μ. From their representations, one can see that $h_{1\mu}$ and $h_{2\mu}$ are differentiable. Their respective subgradients coincides with their gradients, that can be computed by the *partial derivatives* with respect to $x^1, \dots, x^{k+1}$ given by

$$\frac{\partial h_{1\mu}}{\partial x^\ell}(\mathbf{X}) = (1+\lambda)\sum_{i=1}^{m}\left[\frac{x^\ell - a^i}{\mu} - P\left(\frac{x^\ell - a^i}{\mu}; F\right)\right] \text{ for } \ell = 1, \dots, k+1. \tag{5.4}$$

Thus, $\nabla h_{1\mu}(\mathbf{X}))$ is the $(k+1)\times n$ matrix $\mathbf{H}_1$ whose ℓth row is $\frac{\partial h_{21\mu}}{\partial x^\ell}(\mathbf{X})$.

Similarly,

$$\frac{\partial h_{2\mu}}{\partial x^\ell}(\mathbf{X}) = 2\sum_{j=1}^{k+1}\left[\frac{x^\ell - x^j}{\mu} - P\left(\frac{x^\ell - x^j}{\mu}; F\right)\right] \text{ for } \ell = 1, \ldots, k+1. \tag{5.5}$$

Hence, $\nabla h_{2\mu}(\mathbf{X})$ is the $(k+1) \times n$ matrix $\mathbf{H}_4$ whose ℓth row is $\frac{\partial h_{2\mu}}{\partial x^\ell}(\mathbf{X})$.

The procedures for computing a subgradient of h_i for $i = 3, 4, 5$ are similar to those from the previous section. Therefore, we are ready to give a new DCA-based algorithm for the bilevel hierarchical clustering problem in Model II.

Algorithm 3 Model II

```
1: Input: $\mathbf{A}, \mathbf{X}_0, \lambda_0, \mu_0, \sigma_1, \sigma_2, \varepsilon, N \in \mathbb{N}$.
2: while stopping criteria $(\lambda, \mu, \varepsilon)$ = false do
3:     $\alpha := \frac{1}{m(\lambda+1)+2(k+1)}$
4:     $\beta := \frac{2}{m(\lambda+1)[m(\lambda+1)+2(k+1)]}$
5:     for $k = 1, \ldots, N$ do
6:         Find $\mathbf{Y}_k \in \partial h_\mu(\mathbf{X}_{k-1})$
7:         $\mathbf{X}_k = (\alpha\mathbb{I} + \beta\mathbf{E})\left((1+\lambda)\mathbf{EA} + \mu\mathbf{Y}_k\right)$
8:     end for
9:     update $\lambda$ and $\mu$
10: end while
11: Output: $X_N$.
```

Example 5.3 (ℓ^2-clustering with Algorithm 3). In this example, we consider the hierarchical clustering problem in Model II for the case where F is the Euclidean closed unit ball in $\mathbb{R}^n$. To implement Algorithm 3, it remains to find a subgradient $\mathbf{Y} \in \partial h_\mu(\mathbf{X})$. Recall that

$$h_\mu(\mathbf{X}) = h_{1\mu}(\mathbf{X}) + h_{2\mu}(\mathbf{X}) + h_3(\mathbf{X}) + h_4(\mathbf{X}) + h_5(\mathbf{X}) \text{ for } \mathbf{X} \in \mathbb{R}^{(k+1)\times n}.$$

The functions $h_{1\mu}$ and $h_{2\mu}$ are continuously differentiable. The gradients $\nabla h_{1\mu}(\mathbf{X})$ and $\nabla h_{2\mu}(\mathbf{X})$ can be determined by their partial derivatives from (5.4) and (5.5), respectively. We can find subgradients $\mathbf{Y}_3 \in \partial h_3(\mathbf{X})$ and $Y_4 \in \partial h_4(\mathbf{X})$ by the procedure developed in Example 4.5. Now, we focus on finding a subgradient $\mathbf{Y}_5 \in \partial h_5(\mathbf{X})$. In this case,

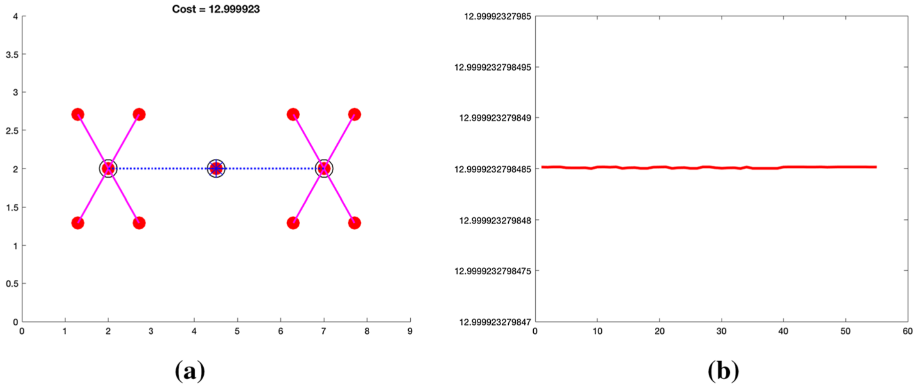
$$h_5(\mathbf{X}) := \max_{t=1,\ldots,k+1} \sum_{\substack{\ell=1 \\ \ell\neq t}}^{k+1}\sum_{j=1}^{k+1} \|x^\ell - x^j\| = \max_{t=1,\ldots,k+1}\left(\sum_{\ell=1}^{k+1}\sum_{j=1}^{k+1}\|x^\ell - x^j\| - \sum_{j=1}^{k+1}\|x^t - x^j\|\right).$$

To find such a subgradient, we will apply the subdifferential sum rule and maximum rule. Choose an index t^* such that

$$\max_{t=1,\ldots,k+1}\left(\sum_{\ell=1}^{k+1}\sum_{j=1}^{k+1}\|x^\ell - x^j\| - \sum_{j=1}^{k+1}\|x^t - x^j\|\right) = \sum_{\ell=1}^{k+1}\sum_{j=1}^{k+1}\|x^\ell - x^j\| - \sum_{j=1}^{k+1}\|x^{t^*} - x^j\|.$$

Define

$$v_{\ell j} := \begin{cases} \frac{x^\ell - x^j}{\|x^\ell - x^j\|} & \text{if } x^\ell \neq x^j, \\ 0 & \text{otherwise.} \end{cases}$$



(a) **(b)**

Fig. 1 Performance of Algorithm 2. **a** Artificial test dataset for Model I, **b** 100% convergence to a global optimal solution

Then $\mathbf{Y}_5$ can be determined by the $(k+1) \times n$ matrix whose ℓth row is given by

$$Y_\ell := 2 \sum_{j=1}^{k+1} v_{\ell j} - v_{\ell t^*} \quad \text{for } \ell = 1, \ldots, k+1.$$

By the procedure developed in Example 4.6 with the use of a *signed matrix*, we can similarly provide another example for hierarchical clustering for Model II in the case where F is the closed unit box in $\mathbb{R}^n$. The detail is left for the reader.

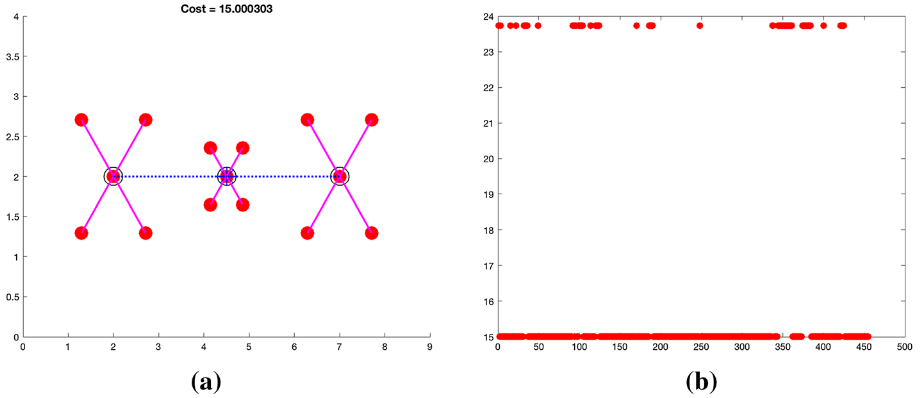
6 Numerical experiments

We conducted our numerical experiments on a MacBook Pro with 2.2 GHz Intel Core i7 Processor, 16 GB 1600 MHz DDR3 Memory (for the MATLAB code), and on a computer having 20 Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz cores and 250 GB RAM (for the C++ code). Even though the two continuous optimization formulations we consider are nonsmooth and nonconvex, Nesterov's smoothing techniques allowed us to design two implementable DCA-based algorithms.

For the implementation of the algorithms, we wrote the codes in MATLAB and C++. Since our algorithms are adaptations of the DCA, there is no guarantee that our algorithms converge to a global optimal solution. However, for the artificial test dataset we created to test the performance of Algorithm 2 with 11 nodes, 2 clearly identifiable cluster centers, and a total center (see Fig. 1), the algorithm converges 100% of the time to a global optimal solution for all 55 different pairs of starting centers selected from the 11 points, i.e., $\binom{11}{2} = 55$.

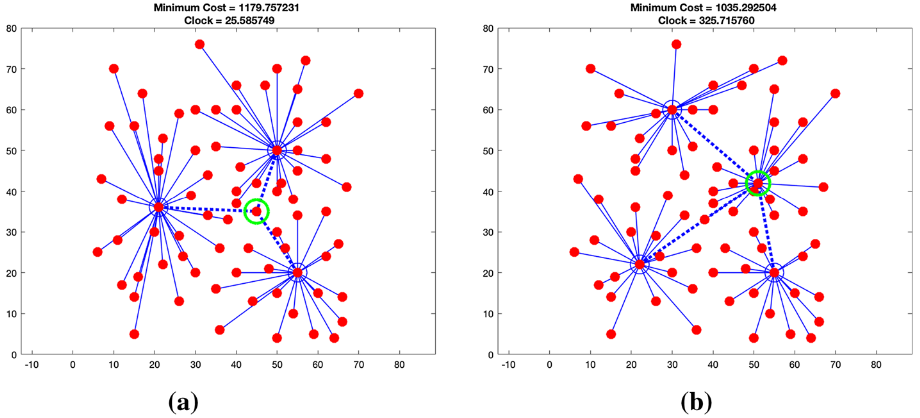
On the other hand, for the artificial test dataset we created to test the performance of Algorithm 2 with 15 nodes, 2 clearly identifiable cluster centers, and a total center (see Fig. 2), the algorithm converges to a global optimal solution 85% of the time, which means that for all 455 different starting centers selected from the 15 points, i.e., $\binom{15}{3} = 455$, the algorithm converges to a global optimal solution 85% of the time.

Further numerical experiments were performed on datasets EIL76 (The 76 City Problem) and PR1002 (The 1002 City Problem) both taken from the Traveling Salesman Problem Library [14], and on datasets 10000RND, 10000RND3D, 1000RND6D, and 100000RND2D



(a) (b)

Fig. 2 Performance of Algorithm 3 on the test dataset. **a** Artificial test dataset for Model II, **b** 85% convergence to a global optimal solution



(a) (b)

Fig. 3 Optimal solutions for Model I and Model II on EIL76. **a** Model I on EIL76, **b** Model II on EIL76

of several sizes and dimensions generated by a uniform random numbers generator. For instance, Figure 3a, b show optimal solutions for Model I and Model II, respectively, for three cluster centers and a total center. The optimal solutions were calculated by the brute-force search method in which we exhaustively generated all the four possible candidates, 3 cluster centers and 1 total center, and then computed the corresponding cost to take the minimum. In this case, we have $\binom{76}{3} = 70{,}300$ combinations for Model I and $\binom{76}{4} = 1{,}282{,}975$ combinations of cluster centers and a total center to check for Model II. For instance, the optimal value for Model I tested on EIL76 with 3 cluster centers and 1 total center is 1179.76, while for Model II with 3 cluster centers and 1 total center, it is 1035.29.

In the two MATLAB codes we wrote to implement the two algorithms, we updated the penalty parameter λ and the smoothing parameter μ in every iteration by the relations $\lambda_{i+1} = \sigma_1 \lambda_i,\ \sigma_1 > 1$, and $\mu_{i+1} = \sigma_2 \mu_i,\ \sigma_2 \in (0, 1)$, respectively. The two parameters were updated until $\mu < 10^{-6}$ (Tables 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11).

For the choice of the starting centers, we used three different methods:

- **Random** We used the "datasample" (a MATLAB built in function) to randomly select starting centers from the existing nodes without replacement.

Table 1 Starting centers selected randomly, MATLAB code

	COST1	COST2	Time1	Time2	Iter1	Iter2	k	m	n
	$\mu_0 = 16,\ \lambda_0 = 0.01,\ \sigma_1 = 160,\ \sigma_2 = 0.5$								
EIL76	1194.29	1048.41	8.04	10.55	1058	1361	3	76	2
EIL76	1201.97	1048.62	6.84	7.84	918	1006	3	76	2
EIL76	1179.76	1041.53	7.31	10.93	986	1413	3	76	2
EIL76	1181.02	1057.87	7.99	7.50	1030	929	3	76	2
EIL76	1208.39	1057.87	6.40	7.57	832	925	3	76	2
EIL76	1179.76	1057.87	8.16	6.77	1030	876	3	76	2
EIL76	1194.29	1091.57	7.89	6.81	1056	881	3	76	2
EIL76	1179.76	1057.87	7.36	7.19	987	927	3	76	2
EIL76	1204.35	1119.50	9.97	9.62	1337	1238	3	76	2
EIL76	1201.97	1054.90	6.98	6.42	928	820	3	76	2

Table 2 Starting centers selected by the k-means, MATLAB code

	COST1	COST2	Time1	Time2	Iter1	Iter2	k	m	n
	$\mu_0 = 16,\ \lambda_0 = 0.01,\ \sigma_1 = 160,\ \sigma_2 = 0.5$								
EIL76	1204.35	1059.01	9.91	6.62	1320	853	3	76	2
EIL76	1179.76	1045.90	7.23	9.29	969	1195	3	76	2
EIL76	1194.29	1049.53	7.84	5.75	1051	738	3	76	2
EIL76	1179.76	1059.01	7.47	6.61	994	853	3	76	2
EIL76	1204.35	1059.01	9.89	6.59	1320	846	3	76	2
EIL76	1179.76	1059.01	7.42	6.64	994	853	3	76	2
EIL76	1181.02	1041.29	7.21	6.18	965	797	3	76	2
EIL76	1201.97	1059.01	6.99	6.57	931	846	3	76	2
EIL76	1181.02	1059.01	7.39	6.62	988	853	3	76	2
EIL76	1201.97	1048.62	6.49	6.67	870	860	3	76	2

Table 3 Starting centers selected randomly, C++ code

	COST1	COST2	Iter1	Iter2	Time1	Time2	k	m	n
	$\mu_0 = 16,\ \lambda_0 = 0.01,\ \sigma_1 = 160,\ \sigma_2 = 0.5$								
EIL76	1224.04	1064.91	952	829	0.09	0.05	3	76	2
EIL76	1195.55	1053.38	1051	874	0.07	0.05	3	76	2
EIL76	1206.92	1041.52	1045	1091	0.07	0.07	3	76	2
EIL76	1206.92	1057.86	1008	855	0.06	0.06	3	76	2
EIL76	1215.56	1065.79	1165	887	0.07	0.05	3	76	2
EIL76	1218.48	1057.86	1263	829	0.07	0.04	3	76	2
EIL76	1197.42	1067.6	988	884	0.04	0.04	3	76	2
EIL76	1206.92	1048.6	1045	1020	0.05	0.04	3	76	2
EIL76	1215.56	1057.86	1148	843	0.05	0.04	3	76	2
EIL76	1215.56	1165.62	1206	920	0.05	0.04	3	76	2

Table 4 Starting centers selected randomly, C++ code

	COST1	COST2	Iter1	Iter2	Time1	Time2	k	m	n
	$\mu_0 = 16,\ \lambda_0 = 0.01,\ \sigma_1 = 160,\ \sigma_2 = 0.5$								
PR1002	2.56341e+06	2.24537e+06	1023	1023	1.31	1	6	1002	2
PR1002	2.16241e+06	1.79317e+06	1023	1023	1.09	1	6	1002	2
PR1002	2.55508e+06	2.25252e+06	1023	1023	1.1	0.99	6	1002	2
PR1002	2.29283e+06	2.12459e+06	1023	1023	1.1	0.99	6	1002	2
PR1002	2.28579e+06	2.02933e+06	1023	1023	1.1	1	6	1002	2
PR1002	2.02867e+06	1.84531e+06	1023	1023	1.1	0.99	6	1002	2
PR1002	2.49236e+06	2.43734e+06	1023	1023	1.1	0.99	6	1002	2
PR1002	3.02324e+06	2.42825e+06	1023	1023	1.1	0.99	6	1002	2
PR1002	2.33796e+06	2.1374e+06	1023	1023	1.1	1	6	1002	2
PR1002	2.37677e+06	1.85446e+06	1023	1023	1.09	1	6	1002	2

Table 5 Starting centers selected randomly, C++ code

	COST1	COST2	Iter1	Iter2	Time1	Time2	k	m	n
	$\mu_0 = 16,\ \lambda_0 = 0.01,\ \sigma_1 = 160,\ \sigma_2 = 0.5$								
10000RND	1.94933e+07	1.8097e+07	1023	1023	11.36	10.1	6	10,000	2
10000RND	2.44543e+07	2.07372e+07	1023	1023	11.15	10.1	6	10,000	2
10000RND	2.36188e+07	1.90255e+07	1023	1023	11.18	10.07	6	10,000	2
10000RND	2.13395e+07	1.81326e+07	1023	1023	11.16	10.09	6	10,000	2
10000RND	1.97625e+07	1.74163e+07	1023	1023	11.17	10.09	6	10,000	2
10000RND	1.9848e+07	1.79588e+07	1023	1023	11.18	10.11	6	10,000	2
10000RND	2.4502e+07	2.0164e+07	1023	1023	11.17	10.08	6	10,000	2
10000RND	2.38836e+07	2.09025e+07	1023	1023	11.16	10.09	6	10,000	2
10000RND	1.81975e+07	1.68355e+07	1023	1023	11.17	10.09	6	10,000	2
10000RND	2.05324e+07	1.68926e+07	1023	1023	11.16	10.1	6	10,000	2

Table 6 Starting centers selected randomly, C++ code

	COST1	COST2	Iter1	Iter2	Time1	Time2	k	m	n
	$\mu_0 = 16,\ \lambda_0 = 0.01,\ \sigma_1 = 160,\ \sigma_2 = 0.5$								
10000RND	5.17176e+06	5.10097e+06	1023	1023	218.72	166.85	100	10,000	2
10000RND	5.32321e+06	5.20111e+06	1023	1023	218.1	164.76	100	10,000	2
10000RND	5.32893e+06	5.21018e+06	1023	1023	215.79	166.91	100	10,000	2
10000RND	5.45463e+06	5.34531e+06	1023	1023	217.58	166.92	100	10,000	2
10000RND	5.59697e+06	5.42149e+06	1023	1023	217.25	164.93	100	10,000	2
10000RND	5.57053e+06	5.39613e+06	1023	1023	215.23	169.07	100	10,000	2
10000RND	5.67843e+06	5.55442e+06	1023	1023	217.15	166.78	100	10,000	2
10000RND	5.7148e+06	5.57767e+06	1023	1023	215.6	165.05	100	10,000	2
10000RND	5.37335e+06	5.28977e+06	1023	1023	219.63	164.81	100	10,000	2
10000RND	5.73865e+06	5.61554e+06	1023	1023	217.23	166.87	100	10,000	2

Table 7 Starting centers selected randomly, C++ code

	COST1	COST2	Iter1	Iter2	Time1	Time2	k	m	n
	$\mu_0 = 16,\ \lambda_0 = 0.01,\ \sigma_1 = 160,\ \sigma_2 = 0.5$								
10000RND3D	2.83948e+07	2.63213e+07	1023	1023	24.79	19.71	10	10,000	3
10000RND3D	2.74404e+07	2.65681e+07	1023	1023	24.6	19.7	10	10,000	3
10000RND3D	2.9869e+07	2.85641e+07	1023	1023	24.59	19.7	10	10,000	3
10000RND3D	3.44097e+07	3.07609e+07	1023	1023	24.6	19.7	10	10,000	3
10000RND3D	3.05076e+07	2.89047e+07	1023	1023	24.6	19.7	10	10,000	3
10000RND3D	2.72841e+07	2.61452e+07	1023	1023	24.61	19.7	10	10,000	3
10000RND3D	2.94171e+07	2.81767e+07	1023	1023	24.6	22.25	10	10,000	3
10000RND3D	3.15467e+07	2.72963e+07	1023	1023	24.6	19.69	10	10,000	3
10000RND3D	2.78719e+07	2.64644e+07	1023	1023	24.61	21.48	10	10,000	3
10000RND3D	2.80267e+07	2.64164e+07	1023	1023	24.59	19.7	10	10,000	3

Table 8 Starting centers selected randomly, C++ code

	COST1	COST2	Iter1	Iter2	Time1	Time2	k	m	n
	$\mu_0 = 16,\ \lambda_0 = 0.01,\ \sigma_1 = 160,\ \sigma_2 = 0.5$								
1000RND6D	5.68343e+06	5.49334e+06	1023	1023	2.72	2.16	10	1000	6
1000RND6D	6.15169e+06	5.94648e+06	1023	1023	2.5	2.15	10	1000	6
1000RND6D	5.95467e+06	5.87668e+06	1023	1023	2.51	2.15	10	1000	6
1000RND6D	5.848e+06	5.67641e+06	1023	1023	2.5	2.16	10	1000	6
1000RND6D	5.82286e+06	5.73382e+06	1023	1023	2.5	2.15	10	1000	6
1000RND6D	5.81637e+06	5.49823e+06	1023	1023	2.51	2.15	10	1000	6
1000RND6D	6.00205e+06	5.84304e+06	1023	1023	2.5	2.15	10	1000	6
1000RND6D	5.9963e+06	5.86284e+06	1023	1023	2.5	2.17	10	1000	6
1000RND6D	6.16517e+06	6.03364e+06	1023	1023	2.5	2.14	10	1000	6
1000RND6D	5.71309e+06	5.60686e+06	1023	1023	2.51	2.15	10	1000	6

Table 9 Starting centers selected randomly, C++ code

	COST1	COST2	Iter1	Iter2	Time1	Time2	k	m	n
	$\mu_0 = 16,\ \lambda_0 = 0.01,\ \sigma_1 = 160,\ \sigma_2 = 0.5$								
100000RND2D	1.40282e+08	1.33498e+08	1023	1023	198.3	165.35	10	100,000	2
100000RND2D	1.83297e+08	1.54512e+08	1023	1023	197.06	168.74	10	100,000	2
100000RND2D	1.5134e+08	1.41451e+08	1023	1023	198.74	165.34	10	100,000	2
100000RND2D	1.59333e+08	1.4203e+08	1023	1023	199.65	164.96	10	100,000	2
100000RND2D	1.53366e+08	1.35764e+08	1023	1023	199.08	167.07	10	100,000	2
100000RND2D	1.55465e+08	1.45342e+08	1023	1023	199.99	166.82	10	100,000	2
100000RND2D	1.39211e+08	1.32843e+08	1023	1023	197.37	165.71	10	100,000	2
100000RND2D	1.60153e+08	1.4911e+08	1023	1023	199.78	167.28	10	100,000	2
100000RND2D	1.52469e+08	1.38242e+08	1023	1023	200.14	167.13	10	100,000	2
100000RND2D	1.46638e+08	1.38241e+08	1023	1023	197.63	165.07	10	100,000	2

Table 10 ℓ^1-norm, Starting centers selected randomly, C++ code, 17 nodes, in 2 dimensions

	COSTL1	Iter1	Time1	k	m	n
	$\mu_0 = 1,\ \lambda_0 = 0.001,\ \sigma_1 = 160,\ \sigma_2 = 0.5$					
17NODES	31.0543	100	0	2	17	2
17NODES	33.9313	100	0	2	17	2
17NODES	50.906	100	0.01	2	17	2
17NODES	29.9189	100	0	2	17	2
17NODES	31.5767	100	0	2	17	2
17NODES	29.8145	100	0.01	2	17	2
17NODES	39.6129	100	0	2	17	2
17NODES	28.9086	100	0	2	17	2
17NODES	31.5767	100	0	2	17	2
17NODES	50.906	100	0	2	17	2

Table 11 ℓ^1-norm, Starting centers selected randomly, C++ code

	COSTL1	Iter1	Time1	k	m	n
	$\mu_0 = 1,\ \lambda_0 = 0.001,\ \sigma_1 = 160,\ \sigma_2 = 0.5$					
10000RND	2.3109e+07	46	1.15	10	10,000	2
10000RND	1.86801e+07	46	0.95	10	10,000	2
10000RND	2.02004e+07	46	0.94	10	10,000	2
10000RND	1.78291e+07	55	1.12	10	10,000	2
10000RND	2.02704e+07	46	0.95	10	10,000	2
10000RND	2.32961e+07	46	0.95	10	10,000	2
10000RND	1.97015e+07	55	1.12	10	10,000	2
10000RND	2.18099e+07	55	1.15	10	10,000	2
10000RND	2.51168e+07	55	1.13	10	10,000	2
10000RND	2.16445e+07	46	0.94	10	10,000	2

- **K-means clustering** We used the “kmeans” (a MATLAB built in function) to partition the nodes into k clusters first, and then we selected the k cluster centroid locations as starting centers.
- **C++ implementation** We implemented the model 1 and model 2 algorithms in C++ and used uniform random numbers generator to generate starting centers. The code was developed using Armadillo library and run on a computer having 20 Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz cores and 250 GB RAM.

7 Conclusion and future research

In this study, we presented two DCA-based algorithms for solving two different bilevel hierarchical clustering problems where the similarity(dissimilarity) measure between two data points (nodes) is given by generalized distances. As special cases of generalized distances, we provided two detailed examples for the ℓ^1 and ℓ^2 norms. We implemented the algorithms with MATLAB and C++ and tested them on different datasets of various sizes and dimensions. We expect that our method used in this paper for solving bilevel hierarchical clustering problems are applicable to solving other nonsmooth nonconvex optimization problems.

Acknowledgements This material is based upon work supported by the AFRL Mathematical Modeling and Optimization Institute. The authors are grateful to both anonymous referees for their helpful comments that allowed us to improve the original presentation.

References

1. An, L.T.H., Belghiti, M.T., Tao, P.D.: A new efficient algorithm based on DC programming and DCA for clustering. J. Glob. Optim. **27**, 503–608 (2007)
2. An, L.T.H., Minh, L.H., Tao, P.D.: New and efficient DCA based algorithms for minimum sum-of-squares clustering. Pattern Recognit. **47**, 388–401 (2014)
3. An, L.T.H., Minh, L.H.: Optimization based DC programming and DCA for hierarchical clustering. Eur. J. Oper. Res. **183**, 1067–1085 (2007)
4. An, L.T.H., Tao, P.D.: Convex analysis approach to D.C. programming: theory, algorithms and applications. Acta Math. Vietnam **22**, 289–355 (1997)
5. Bagirov, A.: Derivative-free methods for unconstrained nonsmooth optimization and its numerical analysis. Investig. Oper. **19**, 75–93 (1999)
6. Bagirov, A., Jia, L., Ouveysi, I., Rubinov, A.M.: Optimization based clustering algorithms in Multicast group hierarchies. In: Proceedings of the Australian Telecommunications, Networks and Applications Conference (ATNAC), Melbourne Australia (published on CD, ISNB 0-646-42229-4) (2003)
7. Bagirov, A., Taheri, S., Ugon, J.: Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems. Pattern Recognit. **53**, 12–24 (2016)
8. Barbosa, G. V., Villas-Boas, S. B., Xavier, A. E.: Solving the two-level clustering problem by hyperbolic smoothing approach, and design of multicast networks. In: Selected Proceedings, WCTR RIO (2013)
9. Hiriart-Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms I, II. Springer, Berlin (1993)
10. Mordukhovich, B.S., Nam, N.M.: An Easy Path to Convex Analysis and Applications. Morgan & Claypool Publishers, San Rafael, CA (2014)
11. Nam, N.M., Rector, R.B., Giles, D.: Minimizing differences of convex functions with applications to facility location and clustering. J. Optim. Theory Appl. **173**, 255–278 (2017)
12. Nam, N.M., Geremew, W., Reynolds, S., Tran, T.: The Nesterov smoothing technique and minimizing differences of convex functions for hierarchical clustering. Optim. Lett. **12**, 455–473 (2018)
13. Nesterov, Y.: Smooth minimization of non-smooth functions. Math. Program. **103**, 127–152 (2005)
14. Reinelt, G.: TSPLIB: a traveling salesman problem library. ORSA J. Comput. **3**, 376–384 (1991)
15. Rockafellar, R.T.: Convex Analysis. Princeton University Press, Princeton, NJ (1970)
16. Tao, P.D., An, L.T.H.: A d.c. optimization algorithm for solving the trust-region subproblem. SIAM J. Optim. **8**, 476–505 (1998)