Federated K-Means Clustering via Dual Decomposition-based Distributed Optimization

Anonymous authors
Paper under double-blind review

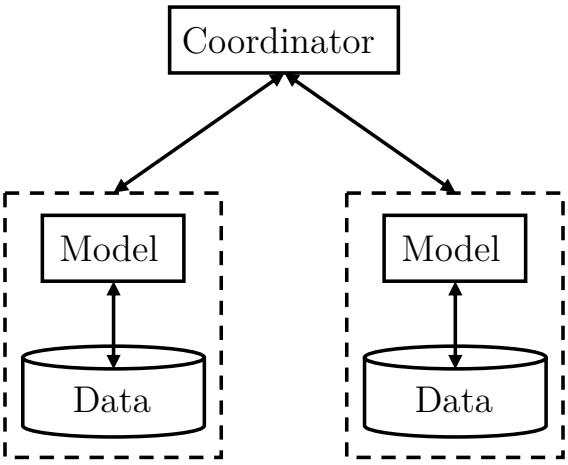
Abstract

The use of distributed optimization in machine learning can be motivated either by the resulting preservation of privacy or the increase in computational efficiency. On the one hand, training data might be stored across multiple devices. Training a global model within a network where each node only has access to its own confidential data requires the use of distributed algorithms. Even if the data is not confidential, sharing it might be prohibitive due to bandwidth limitations. On the other hand, the ever increasing amount of available data leads to large-scale machine learning problems. By splitting the training process across multiple nodes its efficiency can be significantly increased. This paper demonstrates the application of dual decomposition to the distributed training of k-means clustering problems. After an overview of distributed and federated machine learning, the mixed-integer quadratically constrained programming-based formulation of the k-means clustering training problem is presented. The training can be performed in a distributed manner by splitting the data across different nodes and linking these nodes through consensus-constraints. Finally, the performance of the subgradient method, the bundle trust method and the quasi-Newton dual ascent algorithm are evaluated on a set of benchmark problems.

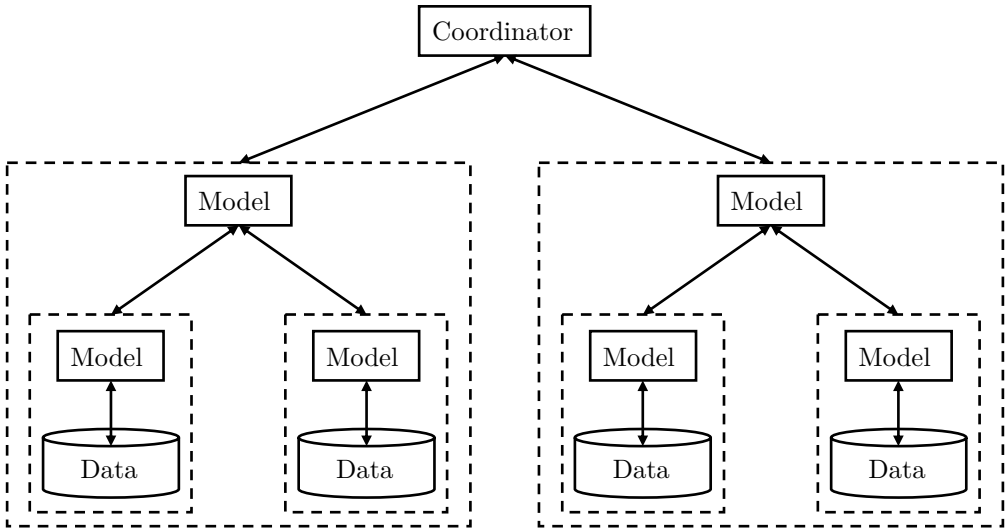
1 Introduction

Training a machine learning model of any kind on a large set of data usually involves the solution of a challenging optimization problem. If the underlying data set becomes too large, it might not be possible to solve the resulting optimization problem in a reasonable amount of time. Distributed optimization methods can aid in rendering the optimization problem tractable through the use of multiple computational resources. Peteiro-Barral & Guijarro-Berdiñas (2013) provide an overview of methods for distributed machine learning . In order to train a global model in a distributed manner a consensus has to be established between the involved nodes and their underlying optimization problems. Forero et al. (2010; 2011) and Georgopoulos & Hasler (2014) demonstrate the distributed training of machine learning models using consensus-based distributed optimization. Tsianos et al. (2012) discuss practical issues with a consensus-based approach which arise from the difference between synchronous and asynchronous communication. Nedić (2020) provides an overview of distributed gradient methods for convex training problems while Verbraeken et al. (2020) give a general survey of distributed machine learning.

While computational performance still remains an issue for many machine learning problems, the increase in computing power and in the efficiency of optimization algorithms can render many challenging problems tractable. However, the inability to share data due to confidentiality reasons still necessitates the use of distributed algorithms. Fig. 1a shows a setting in which training data is stored across two different nodes. Each node can use its local data to train an individual machine learning model. By including a coordination layer the two training processes can be guided in a way that a global model is trained, without the need to share confidential data. If the underlying optimization problems are still hard to solve, the training process can be further divided into subproblems. Fig. 1b depicts the situation in which models of different node clusters are trained in a distributed manner which in turn are again coordinated in order to obtain a global model. Distributed training of a global model without sharing individual training data is often referred to as federated optimization or federated learning (Konečnỳ et al., 2016). Most algorithms for federated learning



(a) Simple architecture for federated learning.



(b) Federated learning of node clusters.

Figure 1: Examples of federated learning architectures.

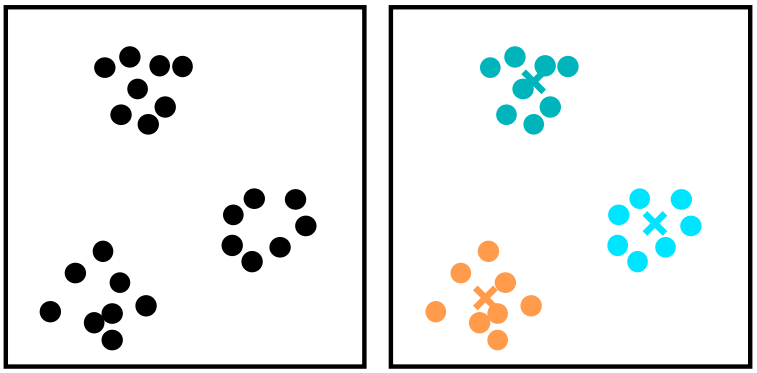
involve an averaging step of the model parameters of the individual nodes (McMahan et al., 2017). Yuan et al. (2021) propose a dual averaging step in order to handle the nonsmoothness of federated composite optimization problems. Federated learning methods have been applied in the context of manufacturing (Hegiste et al., 2022), healthcare (Antunes et al., 2022), mobile devices (Lim et al., 2020) and smart city sensing (Jiang et al., 2020). Li et al. (2020) and Liu et al. (2022) provide surveys on federated learning while Chamikara et al. (2021) examine the privacy aspects related to external attacks. Applying federated learning to heterogeneous data sets can lead to the deterioration of the model quality of individual nodes in regards to their own training data, which might hinder their willingness to participate in such a setting. This issue is addressed through personalized federated learning (Kulkarni et al., 2020; Tan et al., 2022).

2 K-Means clustering

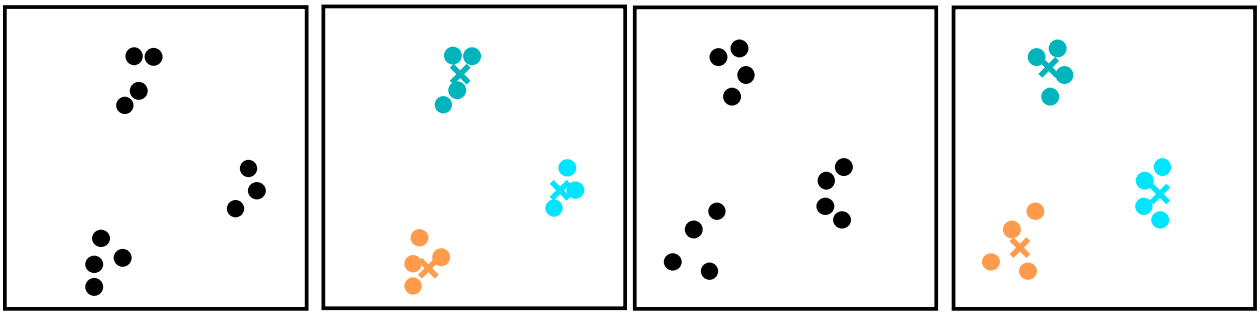
K-means clustering describes an unsupervised machine learning problem in which a set of observations/data is divided into K disjoint clusters according to a similarity measure (Gambella et al., 2021). Clustering problems can be found in many practical application such as image segmentation (Dhanachandra et al., 2015), customer market segmentation (Kansal et al., 2018) or the identification of similar operating points in

a production plant (Rahimi-Adli et al., 2019). This section presents the mixed-integer programming-based formulation of the training problem. The formulation is subsequently extended to the case of distributedly stored data, which gives rise to a federated learning problem. Consensus constraints are used to couple the training problems of different nodes. These constraints can be dualized such that the federated learning problem can be solved via dual decomposition-based distributed optimization. Since the underlying optimization problem contains integrality constraints it is not convex and thus strong duality does not hold. However, a feasible primal solution can be computed in each iteration through an averaging heuristic.

2.1 MIQCP formulation



(a) Clustering of the complete dataset.



(b) Clustering of data-subset 1.

(c) Clustering of data-subset 2.

Figure 2: Illustration of K-means clustering both in a centralized and a decentralized setting.

The goal of K-means clustering is to assign a set of observations $\mathbf{y}_j \in \mathbb{R}^{n_\mathbf{y}}$, $j \in \mathcal{J} = \{1, \dots, |\mathcal{J}|\}$ to a set of clusters $\mathcal{K} = \{1, \dots, K\}$ and to compute the centroids of each cluster. The number of clusters is a hyperparameter and is set a priori or in an iterative manner. This problem can be formulated as a mixed-integer nonlinear programming (MINLP) problem (Aloise et al., 2012; Gambella et al., 2021),

$$\min_{w_{jk}, \mathbf{m}_k} \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} w_{jk} \cdot \|\mathbf{y}_j - \mathbf{m}_k\|_2^2, \tag{1a}$$

$$\text{s.t.} \sum_{k \in \mathcal{K}} w_{jk} = 1, \forall j \in \mathcal{J}, \tag{1b}$$

$$w_{jk} \in \{0, 1\}, \ \forall j \in \mathcal{J}, k \in \mathcal{K}, \ \mathbf{m}_k \in \mathbb{R}^{n_\mathbf{y}} \ \forall k \in \mathcal{K}. \tag{1c}$$

The binary variables w_{jk} indicate if observation $\mathbf{y}_j$ is assigned to cluster k and $\mathbf{m}_k$ is the centroid of cluster k. Constraint (1b) enforces that each observation is assigned to exactly one cluster, while the objective is to minimize the sum of the squared Euclidean distances of all observations to the centroids of their assigned clusters. Problem 1 is a nonconvex MINLP which is hard to solve. In practice it is more efficient to use a linearized formulation by introducing the variable d_{jk}, which describes the squared distance between an

observation j and the centroid of cluster k (Gambella et al., 2021),

$$\min_{w_{jk},d_{jk},\mathbf{m}_k} \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} d_{jk} \tag{2a}$$

$$\text{s.t.} \sum_{k\in\mathcal{K}} w_{jk} = 1, \forall j \in \mathcal{J}, \tag{2b}$$

$$d_{jk} \geq \|\mathbf{y}_j - \mathbf{m}_k\|_2^2 - M_j \cdot (1 - w_{jk}),\ \forall j \in \mathcal{J}, k \in \mathcal{K} \tag{2c}$$

$$w_{jk} \in \{0,1\}, d_{jk} \geq 0,\ \forall j \in \mathcal{J}, k \in \mathcal{K},\ \mathbf{m}_k \in \mathbb{R}^{n_\mathbf{y}}\ \forall k \in \mathcal{K}. \tag{2d}$$

2 is a mixed-integer quadratically constrained programming (MIQCP) problem with a convex integer relaxation. Constraint (2c) is an epigraph formulation of the squared Euclidean distance if observation j is assigned to cluster k, i.e., when $w_{jk} = 1$. Otherwise, the parameter M_j has to be large enough so that the constraint is trivially satisfied for $w_{jk} = 0$. In theory a common big-M parameter can be used for all constraints described by (2c). However, the parameter should be chosen as small as possible in order to avoid weak integer relaxations. In the following the big-M parameter is set as

$$M_j = \max_{\boldsymbol{\chi}\in\mathcal{Y}} \|\mathbf{y}_j - \boldsymbol{\chi}\|_2^2,\ \forall j \in \mathcal{J}, \tag{3a}$$

$$\mathcal{Y} = \{\mathbf{y} \in \mathbb{R}^{n_\mathbf{y}} | \min_{j\in\mathcal{J}} [\mathbf{y}_j]_l \leq [\mathbf{y}]_l \leq \max_{j\in\mathcal{J}} [\mathbf{y}_j]_l,\ l = 1.\ldots, n_\mathbf{y}\}. \tag{3b}$$

Different approaches have been proposed to solve the clustering optimization problem. Bagirov & Yearwood (2006) present a heuristic method based on nonsmooth optimization, Aloise et al. (2012) propose a column generation algorithm and Karmitsa et al. (2017) use a diagonal bundle method. Fig. 2a illustrates the concept of K-means clustering. The unlabeled data (left) is split into 3 clusters according to their distance to the computed cluster centroid (crosses).

2.2 Distributed consensus formulation

Problem 2 describes the case in which the entire data set is accessible from a single node. However, this might not always be the case, especially if the underlying data is confidential. In the following it is assumed that the data set is split across several nodes $\mathcal{I} = \{1, \ldots, N_s\}$, with each node having access to the data-subset $\mathcal{J}_i \subset \mathcal{J}$. The MIQCP problem 2 can be extended to the case of multiple nodes,

$$\min_{w_{ijk},d_{ijk},\mathbf{m}_k} \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}_i}\sum_{k\in\mathcal{K}} d_{ijk} \tag{4a}$$

$$\text{s.t.} \sum_{k\in\mathcal{K}} w_{ijk} = 1, \forall i \in \mathcal{I}, j \in \mathcal{J}_i, \tag{4b}$$

$$d_{ijk} \geq \|\mathbf{y}_j - \mathbf{m}_k\|_2^2 - M_j \cdot (1 - w_{ijk}),\ \forall i \in \mathcal{I}, j \in \mathcal{J}_i, k \in \mathcal{K}, \tag{4c}$$

$$w_{ijk} \in \{0,1\}, d_{ijk} \geq 0,\ \forall i \in \mathcal{I}, j \in \mathcal{J}_i, k \in \mathcal{K},\ \mathbf{m}_k \in \mathbb{R}^{n_\mathbf{y}}\ \forall, k \in \mathcal{K}. \tag{4d}$$

The goal of problem 4 is again to compute a set of cluster centroids $\mathbf{m}_k$ and to assign the observations of all nodes to these clusters. However, if the nodes cannot share their data, problem 4 cannot be solved in a centralized manner. A simple distributed approach would be to solve a clustering problem in each node i. This could lead to situation as depicted in Fig. 2b and Fig. 2c. If each the data-set is split across two nodes, each one can solve a clustering problem. However, both nodes will compute different cluster centroids.
The goal of a federated learning approach is to train a global model, i.e., global cluster centroids in the case of K-means clustering, without sharing the local data between the nodes. To this end each node i can compute individual cluster centroids $\mathbf{m}_{ik}$,

$$\min_{w_{ijk},d_{ijk},\mathbf{m}_{ik}} \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}_i}\sum_{k\in\mathcal{K}} d_{ijk} \tag{5a}$$

$$\text{s.t.} \sum_{k\in\mathcal{K}} w_{ijk} = 1, \forall i \in \mathcal{I}, j \in \mathcal{J}_i, \tag{5b}$$

$$d_{ijk} \geq \|\mathbf{y}_j - \mathbf{m}_{ik}\|_2^2 - M_j \cdot (1 - w_{ijk}),\ \forall i \in \mathcal{I}, j \in \mathcal{J}_i, k \in \mathcal{K}, \tag{5c}$$

$$\mathbf{m}_{ik} = \mathbf{m}_{i'k},\ \forall i \in \mathcal{I}, i' \in \mathcal{N}_i, k \in \mathcal{K}, \tag{5d}$$

$$w_{ijk} \in \{0,1\}, d_{ijk} \geq 0,\ \forall i \in \mathcal{I}, j \in \mathcal{J}_i, k \in \mathcal{K},\ \mathbf{m}_{ik} \in \mathbb{R}^{n_\mathbf{y}}\ \forall, i \in \mathcal{I}, k \in \mathcal{K}. \tag{5e}$$

Since the goal is to obtain global cluster centroids, the individual cluster centroids are coupled through consensus constraints (5d), where $\mathcal{N}_i$ contains the set of neighboring nodes of node i. Problem 5 describes a set of N_s subproblems coupled through the consensus constraints. In the following subsection dual variables are used in order to decouple the clustering problems of the different nodes.

3 Dual decomposition-based distributed clustering

This section presents how the consensus formulation (5) of the clustering problem can be decomposed by introducing dual variables. Dual decomposition can be applied to constraint-coupled optimization problems of the form

$$\min_{\mathbf{x}} \sum_{i \in \mathcal{I}} f_i(\mathbf{x}_i), \tag{6a}$$

$$\text{s. t.} \sum_{i \in \mathcal{I}} \mathbf{A}_i \mathbf{x}_i = \mathbf{b}, \tag{6b}$$

$$\mathbf{x}_i \in \mathcal{X}. \tag{6c}$$

Equation (6) describes an optimization problem consisting of of a set of $\mathcal{I} = \{1, \ldots, N_s\}$ subproblems. The subproblems are coupled through the constraints (6b) and each one is described by individual variables $\mathbf{x}_i$ and constraints $\mathcal{X}_i$. Dual decomposition is based on the introduction of dual variables for the coupling constraints (6b) and on the solution of the resulting dual optimization problem. The idea was first introduced by Everett (1963) for problems involving shared limited resources. Problem 5 can also be rewritten as a general constraint-coupled optimization problem by defining the matrix $\mathbf{A}$ describing the connections between the different nodes. In the following only linear network topologies as depicted in Fig. 3 are considered. Note that the discussion in the remainder of this paper can be easily extended to different network topologies.

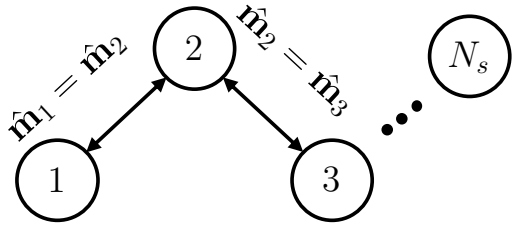


Figure 3: Illustration of a linear network topology and the resulting consensus constraints.

By defining the vector of stacked cluster centroids of each node i,

$$\hat{\mathbf{m}}_i \coloneqq \begin{bmatrix} \mathbf{m}_{i,1} \\ \vdots \\ \mathbf{m}_{i,k} \end{bmatrix} \in \mathbb{R}^{K \cdot n_\mathbf{y}}, \tag{7}$$

the consensus constraints can be rewritten as

$$\hat{\mathbf{m}}_1 - \hat{\mathbf{m}}_2 = \mathbf{0}, \tag{8a}$$

$$\hat{\mathbf{m}}_2 - \hat{\mathbf{m}}_3 = \mathbf{0}, \tag{8b}$$

$$\vdots$$

$$\hat{\mathbf{m}}_{N_s-1} - \hat{\mathbf{m}}_{N_s} = \mathbf{0}. \tag{8c}$$

Constraints 8 can subsequently be rewritten in matrix form

$$\underbrace{\begin{bmatrix} \mathbf{I} & -\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & -\mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} & -\mathbf{I} \end{bmatrix}}_{=:\mathbf{A}\in\mathbb{R}^{K\cdot n_{\mathbf{y}}\cdot(N_s-1)\times K\cdot n_{\mathbf{y}}\cdot N_s}} \cdot \begin{bmatrix} \hat{\mathbf{m}}_1 \\ \hat{\mathbf{m}}_2 \\ \hat{\mathbf{m}}_3 \\ \vdots \\ \hat{\mathbf{m}}_{N_s} \end{bmatrix} = \mathbf{0}, \tag{9a}$$

or in a more compact way

$$\sum_{i\in\mathcal{I}} \mathbf{A}_i \hat{\mathbf{m}}_i = \mathbf{0} \tag{10}$$

with $\mathbf{A}_i \in \mathbb{R}^{K\cdot n_{\mathbf{y}}\cdot(N_s-1)\times K\cdot n_{\mathbf{y}}}$. By introducing dual variables $\boldsymbol{\lambda} \in \mathbb{R}^{n_{K\cdot n_{\mathbf{y}}\cdot(N_s-1)}}$ for the consensus constraints 10 the Lagrange function of problem (5) can be defined,

$$\mathcal{L}(w_{ijk}, d_{ijk}, \mathbf{m}_{ik}, \boldsymbol{\lambda}) = \sum_{i\in\mathcal{I}} \underbrace{\left(\sum_{j\in\mathcal{J}_i} \sum_{k\in\mathcal{K}} d_{ijk} + \boldsymbol{\lambda}^T \mathbf{A}_i \hat{\mathbf{m}}_i \right)}_{=:\mathcal{L}_i(w_{ijk}, d_{ijk}, \mathbf{m}_{ik}, \boldsymbol{\lambda})}. \tag{11}$$

The minimization of the Lagrange function for a fixed value of the dual variables $\boldsymbol{\lambda}$ gives the corresponding value of the dual function.

$$d(\boldsymbol{\lambda}) \coloneqq \min_{w_{ijk}, d_{ijk}, \mathbf{m}_{ik}} \sum_{i\in\mathcal{I}} \mathcal{L}_i(w_{ijk}, d_{ijk}, \mathbf{m}_{ik}, \boldsymbol{\lambda}) \tag{12a}$$

$$\text{s.t.} \sum_{k\in\mathcal{K}} w_{ijk} = 1, \forall i \in \mathcal{I}, j \in \mathcal{J}_i, \tag{12b}$$

$$d_{ijk} \geq \|\mathbf{y}_j - \mathbf{m}_{ik}\|_2^2 - M_j \cdot (1 - w_{ijk}),\ \forall i \in \mathcal{I}, j \in \mathcal{J}_i, k \in \mathcal{K}, \tag{12c}$$

$$w_{ijk} \in \{0,1\}, d_{ijk} \geq 0,\ \forall i \in \mathcal{I}, j \in \mathcal{J}_i, k \in \mathcal{K},$$

$$\mathbf{m}_{ik} \in \mathbb{R}^{n_{\mathbf{y}}}\ \forall, i \in \mathcal{I}, k \in \mathcal{K}. \tag{12d}$$

The dual function has two important properties. First, the value of the dual function is always a lower bound on the solution of its corresponding primal problem, in this case, problem (5) (Nocedal & Wright, 2006). The problem of finding the dual variables that result in the best lower bound is referred to as the dual optimization problem,

$$\max_{\boldsymbol{\lambda}} d(\boldsymbol{\lambda}). \tag{13}$$

The resulting dual problem can be solved in a distributed manner by solving the individual clustering problems for the current value of the dual variables,

$$\min_{w_{ijk}, d_{ijk}, \mathbf{m}_{ik}} \mathcal{L}_i(w_{ijk}, d_{ijk}, \mathbf{m}_{ik}, \boldsymbol{\lambda}) \tag{14a}$$

$$\text{s.t.} \sum_{k\in\mathcal{K}} w_{ijk} = 1,\ \forall j \in \mathcal{J}_i, \tag{14b}$$

$$d_{ijk} \geq \|\mathbf{y}_j - \mathbf{m}_{ik}\|_2^2 - M_j \cdot (1 - w_{ijk}),\ \forall j \in \mathcal{J}_i, k \in \mathcal{K}, \tag{14c}$$

$$w_{ijk} \in \{0,1\}, d_{ijk} \geq 0,\ \forall j \in \mathcal{J}_i, k \in \mathcal{K},$$

$$\mathbf{m}_{ik} \in \mathbb{R}^{n_{\mathbf{y}}}\ \forall k \in \mathcal{K}. \tag{14d}$$

Second, the dual function (12) is always concave, regardless whether the primal problem is convex or not (Nocedal & Wright, 2006). Therefore the dual problem (13) is a convex optimization problem. However, the dual function is usually nondifferentiable due to a changing set of active individual constraints, which means that problem (13) is a nonsmooth optimization problem (Yfantis et al., 2023). The following subsections present some algorithms for the solution of the dual problem, namely the subgradient method, the bundle trust method, and the quasi-Newton dual ascent algorithm.

3.1 Subgradient method

Since the dual function is nondifferentiable a gradient cannot be defined for every value of the dual variables. Instead, a subgradient can be used. A vector $\boldsymbol{\xi} \in \mathbb{R}^{n_{\boldsymbol{\chi}}}$ is a subgradient of a concave function $\phi(\boldsymbol{\chi})$ at a point $\boldsymbol{\chi}_0$ if

$$\phi(\boldsymbol{\chi}) \leq \phi(\boldsymbol{\chi}_0) + \boldsymbol{\xi}^T(\boldsymbol{\chi} - \boldsymbol{\chi}) \tag{15}$$

for all $\boldsymbol{\chi} \in \operatorname{dom} \phi$. The set of all subgradients at a point $\boldsymbol{\chi}_0$ comprise the subdifferential $\partial\phi(\boldsymbol{\chi}_0)$ Technically equation (15) defines a supergradient. Nevertheless, the term subgradient is commonly used in the literature for both convex and concave functions.
A subgradient of the dual function for a given value of the dual variables $\boldsymbol{\lambda}^{(t)}$ can be computed by evaluating the coupling constraints (10),

$$\mathbf{g}(\boldsymbol{\lambda}^{(t)}) = \sum_{i \in \mathcal{I}} \mathbf{A}_i \hat{\mathbf{m}}_i(\boldsymbol{\lambda}^{(t)}) \in \partial d(\boldsymbol{\lambda}^{(t)}), \tag{16}$$

where $\hat{\mathbf{m}}_i(\boldsymbol{\lambda}^{(t)})$ are the cluster centroids obtained by solving the individual clustering problems (14).
In the subgradient method the dual variables are updated in each iteration t along the direction of the subgradient (Shor, 2012)

$$\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} + \alpha^{(t)} \mathbf{g}(\boldsymbol{\lambda}^{(t)}), \tag{17}$$

where $\alpha^{(t)}$ is a step size parameter. The step size parameter plays an important role in the convergence of the algorithm. If it is chosen too large the algorithm might diverge, while a too small choice might significantly slow down its convergence. A common choice to adapt the step size over the course of the iterations is

$$\alpha^{(t)} = \alpha^{(0)} / \sqrt{t}, \tag{18}$$

with an initial step size $\alpha^{(0)}$ (Bertsekas, 1999).

3.2 Bundle trust method

The subgradient method usually exhibits a slow rate of convergence, since only using information from the current subgradient may not provide an ascent direction for the algorithm. Bundle methods are generally more efficient by utilizing multiple subgradients from previous iterations (Mäkelä, 2002). To this end the data

$$\mathcal{B}^{(t)} = \{(\boldsymbol{\lambda}^{(l)}, \mathbf{g}(\boldsymbol{\lambda}^{(l)}), d(\boldsymbol{\lambda}^{(l)})) \in \mathbb{R}^{n_{\boldsymbol{\lambda}}} \times \mathbb{R}^{n_{\boldsymbol{\lambda}}} \times \mathbb{R} |\ l = t - \tau + 1, \ldots, t\} \tag{19}$$

is stored in each iteration, where $n_{\boldsymbol{\lambda}}$ denotes the number of dual variables. $\mathcal{B}^{(t)}$ is referred to as a bundle and it contains the dual variables, subgradients and values of the dual function from previous iterations. Since storing all information from all previous iterations might cause memory issues, only data from the previous τ iterations is used.
The idea of bundle methods is to use the collected information to construct a piece wise linear over approximation of the nonsmooth dual function $d(\boldsymbol{\lambda})$, a so-called cutting plane model,

$$\hat{d}^{(t)}(\boldsymbol{\lambda}) \coloneqq \min_{l \in \{t-\tau+1, \ldots, t\}} \{d(\boldsymbol{\lambda}^{(l)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(l)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(l)})\}. \tag{20}$$

The approximation can be written in an equivalent form as

$$\hat{d}^{(t)}(\boldsymbol{\lambda}) = \min_{l \in \{t-\tau+1, \ldots, t\}} \{d(\boldsymbol{\lambda}^{(t)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(l)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}) - \beta^{(l,t)}\}, \tag{21}$$

with the linearization error

$$\beta^{(l,t)} = d(\boldsymbol{\lambda}^{(t)}) - d(\boldsymbol{\lambda}^{(l)}) - \mathbf{g}^T(\boldsymbol{\lambda}^{(l)})(\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^{(l)}),\ \forall l \in \{t - \tau + 1, \ldots, t\}. \tag{22}$$

The update direction of the dual variables can then be computed by solving a direction finding problem

$$\max_{\mathbf{s} \in \mathbb{R}^{n_{\boldsymbol{\lambda}}}} \hat{d}^{(t)}(\boldsymbol{\lambda}^{(t)} + \mathbf{s}), \tag{23a}$$

$$\text{s. t. } \|\mathbf{s}\|_2^2 \leq \alpha^{(t)}, \tag{23b}$$

(23c)

where constraint (23b) represents a trust region. Therefore, this variant of the bundle method is referred to as bundle trust method (BTM). Other variants include proximal bundle methods, where the trust region is replaced by a regularization term in the objective function (Bagirov et al., 2014). Problem (23) is still a nonsmooth optimization problem and can be transformed into a smooth quadratic direction finding problem by using an epigraph formulation,

$$\max_{v\in\mathbb{R},\ \mathbf{s}\in\mathbb{R}^{n_{\boldsymbol{\lambda}}}} \ v, \tag{24a}$$

$$\text{s. t. } \|\mathbf{s}\|_2^2 \leq \alpha^{(t)}, \tag{24b}$$

$$\mathbf{g}^T(\boldsymbol{\lambda}^{(l)})\mathbf{s} - \beta^{(l,t)} \geq v,\ \forall l \in \{t-\tau+1,\dots,t\}. \tag{24c}$$

After computing a direction the dual variables are updated according to

$$\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} + \mathbf{s}^{(t)}. \tag{25}$$

Bundle methods are widely used in machine learning, as nonsmoothnes is encountered in many training problems involving regularization terms (Le et al., 2007). Bundle methods can also be used to solve the clustering problem (2) (Karmitsa et al., 2017). However, note that in this paper the BTM algorithm is used to solve the nonsmooth dual problem (13).

3.3 Quasi-Newton dual ascent

Since the dual function is always concave it can be locally approximated by a quadratic function. Yfantis & Ruskowski (2022) and Yfantis et al. (2023) recently proposed the quasi-Newton dual ascent (QNDA) algorithm that approximates the dual function by a quadratic function,

$$d_B^{(t)}(\boldsymbol{\lambda}) = \frac{1}{2}(\boldsymbol{\lambda}-\boldsymbol{\lambda}^{(t)})^T\mathbf{B}^{(t)}(\boldsymbol{\lambda}-\boldsymbol{\lambda}^{(t)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(k)})(\boldsymbol{\lambda}-\boldsymbol{\lambda}^{(t)}) + d(\boldsymbol{\lambda}^{(t)}). \tag{26}$$

This follows the idea of Newton methods, where the gradient and Hessian of the function are used within the approximation. However, due to the nonsmoothness of the dual function, the gradient and Hessian are not defined for each value of the dual variable. Instead, the gradient is replaced in eq. (26) by the subgradient and the Hessian is approximated by the matrix $\mathbf{B}^{(t)}$. The approximated Hessian can be updated in each iteration using a Broyden-Fletcher-Goldfarb-Shanno (BFGS) update,

$$\mathbf{B}^{(t)} = \mathbf{B}^{(t-1)} + \frac{\mathbf{y}^{(t)}\mathbf{y}^{(t),T}}{\mathbf{y}^{(t),T}\mathbf{s}^{(t)}} - \frac{\mathbf{B}^{(t-1)}\mathbf{s}^{(t)}\mathbf{s}^{(t),T}\mathbf{B}^{(t-1),T}}{\mathbf{s}^{(t),T}\mathbf{B}^{(t-1)}\mathbf{s}^{(t)}}, \tag{27}$$

where

$$\mathbf{s}^{(t)} \coloneqq \boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^{(t-1)} \tag{28}$$

is the variation of the dual variables and

$$\mathbf{y}^{(t)} = \mathbf{g}(\boldsymbol{\lambda}^{(t)}) - \mathbf{g}(\boldsymbol{\lambda}^{(t-1)}) \tag{29}$$

is the variation of the subgradients.
The approximated dual function $d_B(\boldsymbol{\lambda})$ is differentiable, while the actual dual function is nonsmooth. This can lead to significant approximation errors and poor update directions. This issue can be addressed by utilizing the same information as in the BTM algorithm. However, instead of using the bundle to construct an over approximator of the dual function, it is used to further constrain the update of the dual variables,

$$d_B^{(t)}(\boldsymbol{\lambda}^{(t+1)}) \leq d(\boldsymbol{\lambda}^{(l)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(l)})(\boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^{(l)}),\ \forall l \in \{t-\tau+1,\dots,t\}. \tag{30}$$

Constraints (30) are derived from the definition of the subgradient (15). A violation of these constraints would indicate that the updated dual variables $\boldsymbol{\lambda}^{(t+1)}$ are outside the range of validity of the approximated dual function. These constraints are referred to as bundle cuts and they can be summarized as

$$\mathcal{BC}^{(t)} = \{\boldsymbol{\lambda} \in \mathbb{R}^{n_{\boldsymbol{\lambda}}} |\ d_B^{(t)}(\boldsymbol{\lambda}) \leq d(\boldsymbol{\lambda}^{(l)}) + \mathbf{g}^T(\boldsymbol{\lambda}^{(l)})(\boldsymbol{\lambda}-\boldsymbol{\lambda}^{(l)}),\ \forall l \in \{t-\tau+1,\dots,t\}\}. \tag{31}$$

In the QNDA algorithm the dual variables are updated in each iteration by solving the optimization problem

$$\boldsymbol{\lambda}^{(t+1)} = \underset{\boldsymbol{\lambda}}{\operatorname{argmax}}\ d_B^{(t)}(\boldsymbol{\lambda}), \tag{32a}$$

$$\text{s. t. } \|\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(t)}\|_2^2 \leq \alpha^{(t)}, \tag{32b}$$

$$\boldsymbol{\lambda} \in \mathcal{BC}^{(t)}. \tag{32c}$$

$$\tag{32d}$$

To avoid too aggressive update steps the same trust region (32b) as in the BTM algorithm is used.

3.4 Primal heuristics

The following sections provide some additional heuristics related to the primal optimization problem (5), namely an averaging heuristic used to obtain feasible primal solutions, and the addition of symmetry breaking constraints to the clustering problem.

3.4.1 Averaging heuristic

The K-means clustering problem involves integrality constraints and is therefore nonconvex. While the (optimal) value of the dual function 12 provides a lower bound on the optimal value of the primal problem 5, feasibility of the primal problem is not guaranteed upon convergence of a dual decomposition-based algorithm, i.e., the consensus constraints may not be satisfied. Nevertheless, in the case of K-means clustering it is straightforward to compute a feasible primal solution using an averaging step. In each iteration t of a dual decomposition-based algorithm the coordinator communicates the dual variables $\boldsymbol{\lambda}^{(t)}$ to the nodes. The nodes in turn solve their individual clustering problems and communicate their computed cluster centroids $\hat{\mathbf{m}}_i(\boldsymbol{\lambda}^{(t)})$ to the coordinator. Based on this response the coordinator can compute the average of the primal variables, i.e., the average cluster centroids,

$$\overline{\mathbf{m}}_k(\boldsymbol{\lambda}^{(t)}) = \frac{1}{N_s} \sum_{i \in \mathcal{I}} \mathbf{m}_{ik}(\boldsymbol{\lambda}^{(t)}) \tag{33}$$

which are then communicated back to the nodes. Using the mean cluster centroids the nodes can compute their resulting primal objective value

$$z_i(\boldsymbol{\lambda}^{(t)}) = \sum_{j \in \mathcal{J}} \min_{k \in \mathcal{K}} \|\mathbf{y}_j - \overline{\mathbf{m}}_k(\boldsymbol{\lambda}^{(t)})\|_2^2. \tag{34}$$

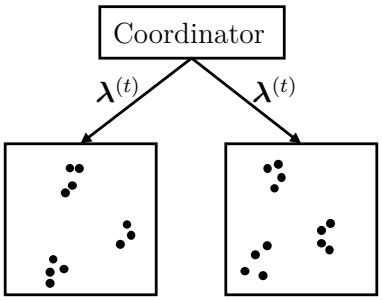
The primal objective value can be used to compute the relative duality gap in each iteration,

$$\text{rel. DG} = 100 \cdot \left(1 - \frac{d(\boldsymbol{\lambda}^{(t)})}{\sum_{i \in \mathcal{I}} z_i(\boldsymbol{\lambda}^{(t)})}\right). \tag{35}$$

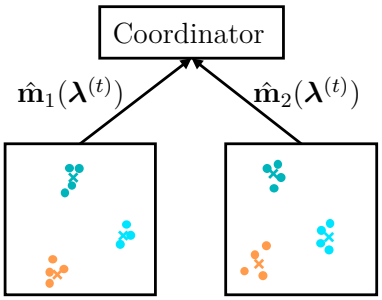
Since the value of the dual function provides a lower bound on the optimal primal objective value the relative duality gap can be used to assess the distance of a found solution to the global optimum. The entire communication process between the coordinator and the nodes is illustrated in Fig. 4. Note that the average cluster centroids are only used to compute the duality gap. They do not influence the update of the dual variables.

3.4.2 Symmetry breaking constraints

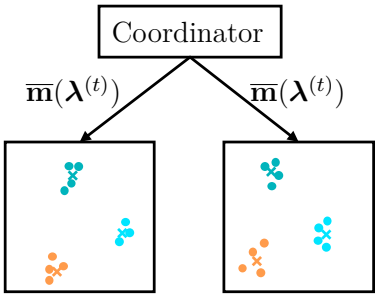
The clustering problem 4 is highly symmetric, i.e., it contains solutions with the same objective values. This is due to the fact that the index assigned to a cluster does not influence the objective function. Fig. 5 illustrates the situation of two symmetric solutions. This symmetry can lead to problems for the averaging heuristic presented in the previous section, as the computed cluster centroids of a single node can switch from one iteration to the next. For instance, while some points are assigned to cluster k in iteration t, they



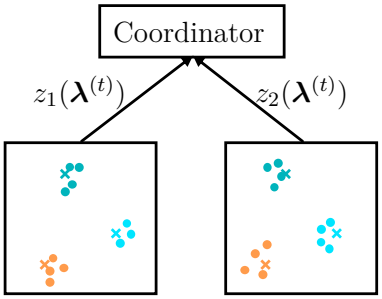
(a) The coordinator sends the current dual variables to the nodes.



(b) The nodes compute their cluster centroids and send them to the coordinator.



(c) The coordinator computes the average cluster centroids and sends them to the nodes.



(d) The nodes compute their objectives based on the received average centroids.

Figure 4: Communication process between the coordinator and the nodes in iteration t.

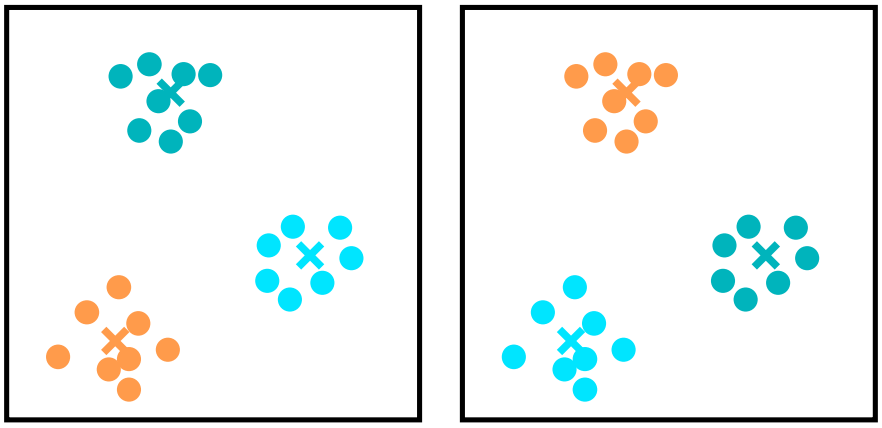


Figure 5: Example of symmetric clustering solutions. In the two cases the data points are assigned to different clusters without affecting the objective function.

could be assigned to cluster k' in iteration $t+1$ by switching the centroids of clusters k and k' without affecting the objective.

In order to prevent this behavior symmetry breaking constraints are added to the optimization problems of the nodes. In the first iteration one of the nodes acts as the reference nodes, providing reference centroids $\overline{\mathbf{m}}_k^{\text{ref}}$. In the subsequent iterations the quadratic constraint

$$\|\mathbf{m}_{ik} - \overline{\mathbf{m}}_k^{\text{ref}}\|_2^2 \leq \|\mathbf{m}_{ik'} - \overline{\mathbf{m}}_k^{\text{ref}}\|_2^2, \forall k, k' \in \mathcal{K}, \tag{36}$$

is added to each node i. This ensures that cluster k of each node i will be the one closest to the reference centroid $\overline{\mathbf{m}}_k^{\text{ref}}$. The choice of the node which provides the reference centroid can be performed arbitrarily, as it does not affect the optimization of the other nodes. Furthermore, the added constraint also does not affect the optimal objective value while rendering all symmetric solutions, except for one, infeasible.

4 Numerical analysis of distributed clustering problems

The dual decomposition-based distributed clustering approach was evaluated on a set of benchmark problems of varying size. The data for each benchmark problem was generated randomly. First, initial cluster centroids $\mathbf{m}_k^0$ were generated, with $[\mathbf{m}_k^0]_l \in \mathcal{U}_c(-1, 1)$, $l = 1, \ldots, n_\mathbf{y}$. Then, for each cluster k five random data points where added within a radius of 0.5 from the generated centroid. The parameters of the benchmark problems were varied as follows:

$$\begin{aligned}&\text{Number of nodes: } N_s \in \{2, 3, 4\},\\ &\text{Number of dimensions: } n_\mathbf{y} \in \{2, 3, 4\},\\ &\text{Number of clusters: } K \in \{3, 4\}.\end{aligned}$$

Five benchmark problems were generated for each combination of nodes, dimensions and clusters, resulting in a total of 90 benchmark problems. A benchmark problem is characterized by its number of nodes, dimension of the data and number of clusters. For instance, problem $3N2D4K_5$ is the 5th benchmark problem comprised of 3 nodes with 2-dimensional data sorted into 4 clusters.
The benchmark problems were solved using the subgradient method, the bundle trust method and the quasi-Newton dual ascent algorithm. The use of ADMM was omitted for several reasons. First, in each communication round a feasible primal solution is obtained through the averaging heuristic (cf. Section 3.4.1). This primal solution does not correspond to the current dual variables. Due to the nonconvexity of the underlying MIP problem no guarantee can be made that the consensus constraints will be satisfied at the dual optimum. This in turn means that the regularization term in ADMM might not vanish, which would result in different objective values of the Lagrangian and the augmented Lagrangian, leading to an overestimation of the dual value and a subsequent underestimation of the duality gap, Second, the regularization of ADMM introduces a bias towards the mean cluster centroids. Note that the averaging heuristic does not affect the iterations of the other distributed optimization algorithms. It merely serves to compute a feasible primal solution, i.e., an upper objective bound, in each iteration. Introducing a bias towards the mean centroids in the solution of the clustering problems of the nodes would result in a stagnation of the algorithm. For badly chosen regularization parameters all nodes would converge towards the initial mean centroids, which is not the case in general for the other algorithms. The QADA algorithm could also be used to solve the clustering problem. However, the numerical tests showed that the BTM and QNDA algorithms are already efficient enough to converge within the sampling phase of QADA. Its inclusion in the results was therefore omitted.

Table 1: Parameter settings of the distributed optimization algorithms for the clustering benchmark problems.

	Value	**Description**	**Algorithms**
$\boldsymbol{\lambda}^{(0)}$	$\mathbf{0}$	initial dual variables	All
$\alpha^{(0)}$	0.5	initial step size/trust region parameter	All
$t_{\max}$	150	maximum number of iterations	All
ϵ_p	10^{-2}	primal residual convergence tolerance	All
ϵ_{DG}	0.25 %	relative duality gap tolerance	All
ϵ_b	1	bundle cuts threshold	QNDA
τ	50	allowed age of data points	BTM, QNDA
$\mathbf{B}^{(0)}$	$-\mathbf{I}$	initial approximated Hessian	QNDA

The initial step size (SG)/ trust region (BTM, QNDA) parameter was set to $\alpha^{(0)} = 0.5$ and varied according to

$$\alpha^{(t)} = \alpha^{(0)}/\sqrt{t}. \tag{37}$$

The bundle cuts for QNDA were used in every iteration, i.e., $\epsilon_b = 1$ and $\tau = 50$ points were used to construct the bundle in BTM and the bundle cuts in QNDA respectively. All algorithms were initialized with $\boldsymbol{\lambda}^{(0)} = \mathbf{0}$ and the initial approximated Hessian of the QNDA algorithm was set to the negative identity matrix. The

algorithms were terminated either when the Euclidean norm of the primal residual

$$\|\mathbf{w}_p\|_2 = \left\| \sum_{i \in \mathcal{I}} \mathbf{A}_i \hat{\mathbf{m}}_i \right\|_2 , \tag{38}$$

i.e., the violation of the consensus constraints lied below a threshold of $\epsilon_p = 10^{-2}$ or when the relative duality gap 35 reached a value of $\epsilon_{DG} = 0.25$ %. The used parameters for the different algorithms are summarized in Tab. 1. The MIQCP clustering problems of all nodes were solved using the commercial solver Gurobi and the total computation time was obtained through eq. **??**, with $T_{\text{comm}} = 800\ ms$.
The results for the clustering benchmarks are summarized in Tab. 2.

Table 2: Summary of the results for the distributed optimization of the clustering benchmark problems, $\bar{t}$: mean number of iterations until termination, $\overline{\text{rel. DG}}$: mean relative duality gap upon termination (in %), $\overline{T_{\text{comp}}}$: mean computation time (in s).

Algorithm	$\bar{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$
SG	136.75	2.27	996.28
BTM	57.44	1.86	515.77
QNDA	**54.48**	**1.81**	**483.22**

Out of the examined algorithms, QNDA shows the best performance in terms of the required number of iterations and computation time as well as in terms of the achieved relative duality gap. The BTM algorithm shows a similar performance in terms of the number of iterations and the achieved duality gap. However, in the case of distributed clustering each iteration is costly due to the underlying MIQCP problems. Therefore, a slight performance increase in the number of iterations results in a more substantial performance increase in terms of computation times. More detailed results for the clustering benchmarks are summarized in Tab. 3 in the appendix.

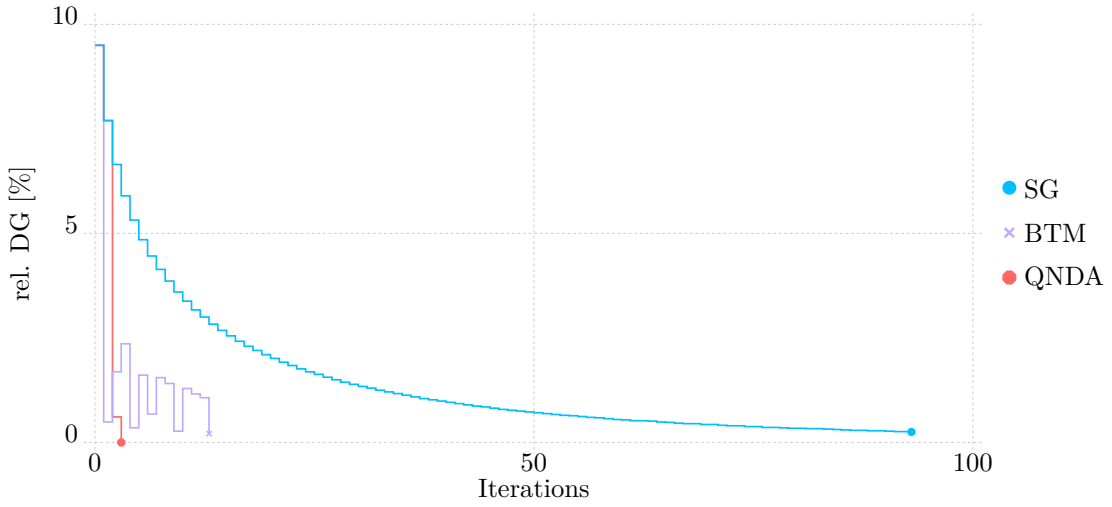
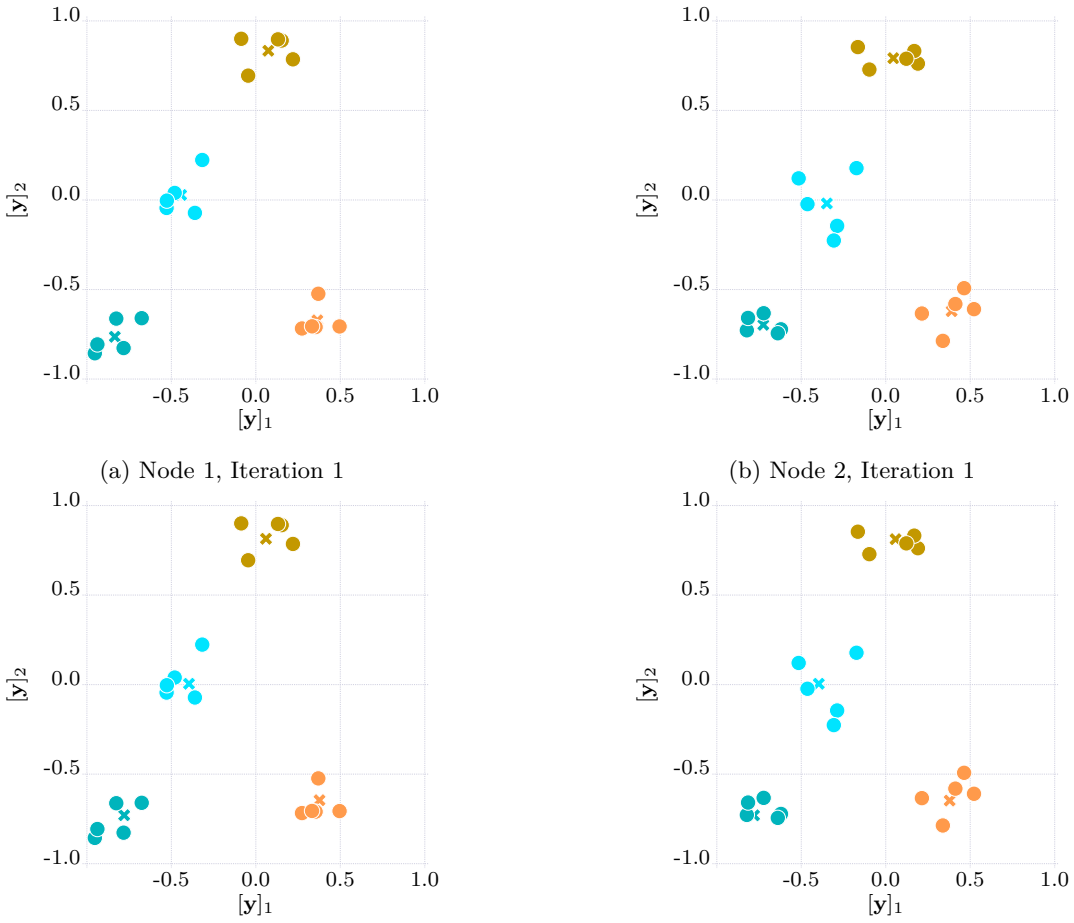


Figure 6: Evolution of the relative duality gap for benchmark problem 2N2D4K$_3$.

Fig. 6 shows the evolution of the relative duality gap for benchmark problem 2N2D4K$_3$. The subgradient method converges rather slowly. In comparison, the BTM and QNDA algorithms exhibit a faster rate of convergence. Between these two algorithms, BTM exhibits an oscillatory behavior before converging. In contrast, the QNDA algorithm does not exhibit oscillations and therefore converges earlier. Additionally, it should be noted that the QNDA algorithm achieves a relative duality gap of 0 %, i.e., it converges to a proven global optimum.



(a) Node 1, Iteration 1

(b) Node 2, Iteration 1

(c) Node 1, Iteration 4

(d) Node 2, Iteration 4

Figure 7: Exemplary clusters in different iterations of the QNDA algorithm for benchmark problem 2N2D4K$_3$.

Fig. 7 provides some further illustrations of the results. Fig. 7a and Fig. 7b show the results of the clustering in the first iteration, i.e., the individual global optima. Fig. 7c and Fig. 7d depict the solutions upon convergence of the QNDA algorithm. It can be seen that each node computes the same cluster centroids corresponding to the globally optimal solution with respect to the entire data set, but not to the individual data sets. It is therefore possible to compute a global model locally in each node while only accessing local data.

5 Comparison to the central solution

As shown in the previous section, the solution of the MIQCP clustering problems is computationally expensive. This is due to the weak integer relaxation of problem 2, which means that the solution of the relaxed problem within the branch-and-bound algorithm is far away from the integer solution. This results in slow moving relative integrality gaps and to slow convergence of the solution algorithm. While the main motivation of the distributed clustering approach is the training of a global model without exchange of local data, it can also be used to efficiently solve larger clustering problems. Fig. 8 depicts the evolution of the relative duality gap of the QNDA algorithm as well as the evolution of the relative integrality gap of Gurobi for the complete data set of benchmark problem 4N4D4K$_3$. The clustering problems of the individual nodes were solved in a sequential manner in the case of QNDA, also using Gurobi. While the relative gap of the central solution improves very slowly, the QNDA algorithm quickly converges to a solution close to the global

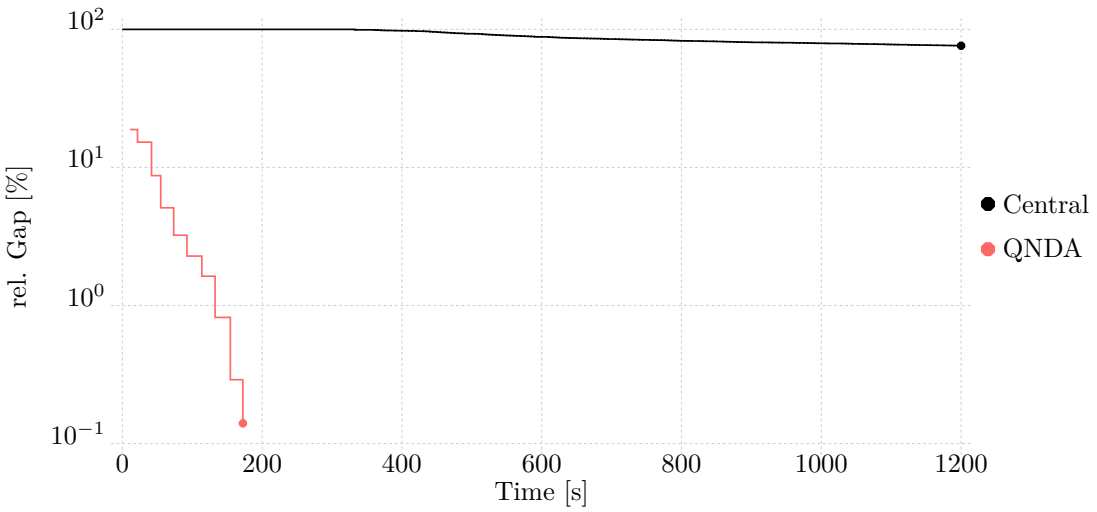


Figure 8: Evolution of the relative duality gap of QNDA compared to the relative integrality gap of the central solution using Gurobi for benchmark problem 4N4D4K$_3$.

optimum. Note, that both relative gaps prove a worst-case distance to the global optimum. Hence, decomposing a large clustering problem into smaller subproblems and coordinating the solutions via a distributed optimization algorithm can offer significant performance improvements compared to a central solution.

6 Conclusion

This paper demonstrated how dual decomposition-based distributed optimization can be applied to the solution of clustering problems. The approach ensures privacy, i.e., enables federated learning, as each node only has access to its local data. A global model can still be obtained by coordinating the solutions of the individual clustering problems. Numerical tests on a large set of benchmark problems demonstrated that the QNDA algorithm outperforms the subgradient method and the BTM algorithm. Furthermore, the distributed optimization approach exhibited superior performance compared to a central solution approach. In the future the developed algorithms can also be applied to other federated learning problems, like the distributed training of support vector machines.

References

D. Aloise, P. Hansen, and L. Liberti. An improved column generation algorithm for minimum sum-of-squares clustering. *Mathematical Programming*, 131:195–220, 2012.

R.S. Antunes, C. André da Costa, A. Küderle, I.A. Yari, and B. Eskofier. Federated learning for healthcare: Systematic review and architecture proposal. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–23, 2022.

A. Bagirov, N. Karmitsa, and M. Mäkelä. *Introduction to Nonsmooth Optimization: Theory, Practice and Software*. Springer, 2014.

A.M Bagirov and J. Yearwood. A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. *European Journal of Operational Research*, 170(2):578–596, 2006.

D. P. Bertsekas. *Nonlinear programming*. Athena Scientific, 1999.

M.A.P. Chamikara, P. Bertok, I. Khalil, D. Liu, and S. Camtepe. Privacy preserving distributed machine learning with federated learning. *Computer Communications*, 171:112–125, 2021.

N. Dhanachandra, K. Manglem, and Y.J. Chanu. Image segmentation using k-means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, 54:764–771, 2015.

H. Everett. Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, (11 (3)):399–417, 1963.

P.A. Forero, A. Cano, and G.B Giannakis. Consensus-based distributed support vector machines. *Journal of Machine Learning Research*, 11(5), 2010.

P.A. Forero, A. Cano, and G.B. Giannakis. Distributed clustering using wireless sensor networks. *IEEE Journal of Selected Topics in Signal Processing*, 5(4):707–724, 2011.

C. Gambella, B. Ghaddar, and J. Naoum-Sawaya. Optimization problems for machine learning: A survey. *European Journal of Operational Research*, 290(3):807–828, 2021.

L. Georgopoulos and M. Hasler. Distributed machine learning in networks by consensus. *Neurocomputing*, 124:2–12, 2014.

V. Hegiste, T. Legler, and M. Ruskowski. Application of federated machine learning in manufacturing. In *2022 International Conference on Industry 4.0 Technology (I4Tech)*, pp. 1–8. IEEE, 2022.

J.C. Jiang, B. Kantarci, S. Oktug, and T. Soyata. Federated learning in smart city sensing: Challenges and opportunities. *Sensors*, 20(21):6230, 2020.

T. Kansal, S. Bahuguna, V. Singh, and T. Choudhury. Customer segmentation using k-means clustering. In *International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, pp. 135–139. IEEE, 2018.

N. Karmitsa, A.M. Bagirov, and S. Taheri. New diagonal bundle method for clustering problems in large data sets. *European Journal of Operational Research*, 263(2):367–379, 2017.

J. Konečnỳ, B. McMahan, D. Ramage, and P. Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

V. Kulkarni, M. Kulkarni, and A. Pant. Survey of personalization techniques for federated learning. In *4th World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pp. 794–797. IEEE, 2020.

Q. Le, A. Smola, and S. Vishwanathan. Bundle methods for machine learning. *Advances in Neural Information Processing Systems*, 20, 2007.

L. Li, Y. Fan, M. Tse, and K.-Y. Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

W.Y.B. Lim, N.C. Luong, D.T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3):2031–2063, 2020.

J. Liu, J. Huang, Y. Zhou, X. Li, S. Ji, H. Xiong, and D. Dou. From distributed machine learning to federated learning: A survey. *Knowledge and Information Systems*, 64(4):885–917, 2022.

M. Mäkelä. Survey of bundle methods for nonsmooth optimization. *Optimization Methods and Software*, 17 (1):1–29, 2002. ISSN 1055-6788. doi: 10.1080/10556780290027828.

B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

A. Nedić. Distributed gradient methods for convex machine learning problems in networks: Distributed optimization. *IEEE Signal Processing Magazine*, 37(3):92–101, 2020.

J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

D. Peteiro-Barral and B. Guijarro-Berdiñas. A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2:1–11, 2013.

K. Rahimi-Adli, P.D. Schiermoch, B. Beisheim, S. Wenzel, and S. Engell. A model identification approach for the evaluation of plant efficiency. In *Computer Aided Chemical Engineering*, volume 46, pp. 913–918. Elsevier, 2019.

N.Z. Shor. *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media, 2012.

A.Z. Tan, H. Yu, L. Cui, and Q. Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

K.I. Tsianos, S. Lawlor, and M.G. Rabbat. Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning. In *50th Annual Allerton Conference on Communication, Control, and Computing*, pp. 1543–1550. IEEE, 2012.

J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J.S. Rellermeyer. A survey on distributed machine learning. *ACM Computing Surveys*, 53(2):1–33, 2020.

V. Yfantis and M. Ruskowski. A hierarchical dual decomposition-based distributed optimization algorithm combining quasi-Newton steps and bundle methods. In *30th Mediterranean Conference on Control and Automation (MED)*, pp. 31–36. IEEE, 2022.

V. Yfantis, S. Wenzel, A. Wagner, M. Ruskowski, and S. Engell. Hierarchical distributed optimization of constraint-coupled convex and mixed-integer programs using approximations of the dual function. *EURO Journal on Computational Optimization*, 11:100058, 2023.

H. Yuan, M. Zaheer, and S. Reddi. Federated composite optimization. In *International Conference on Machine Learning*, pp. 12253–12266. PMLR, 2021.

A Results for the clustering benchmark problems

Table 3: Results for the distributed optimization of the clustering benchmark problems, $\overline{t}$: mean number of performed iterations, $\overline{\text{rel. DG}}$: mean relative duality gap (in %), $\overline{T_{\text{comp}}}$: mean computation time (in s).

	SG			BTM		
Clustering	$\overline{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$	$\overline{t}$	$\overline{\text{rel. DG}}$	$\overline{T_{\text{comp}}}$
Mean	**136.75**	**2.27**	**996.28**	**57.44**	**1.86**	**515.77**
2N2D3K	126.0	1.94	166.08	68.0	1.84	95.01
2N2D4K	113.2	0.89	431.02	64.4	0.8	348.56
2N3D3K	120.0	1.8	223.69	71.6	1.6	167.72
2N3D4K	115.6	0.27	782.18	13.6	0.1	93.92
2N4D3K	91.6	0.31	184.21	36.2	0.16	108.0
2N4D4K	90.4	0.25	965.26	8.6	0.08	93.7
3N2D3K	138.4	7.01	404.59	123.2	6.14	424.47
3N2D4K	150.0	4.7	879.48	62.8	3.99	751.33
3N3D3K	150.0	2.33	301.63	67.0	1.76	160.05
3N3D4K	150.0	0.82	1469.97	66.6	0.35	906.26
3N4D3K	150.0	2.07	354.48	37.2	1.63	110.82
3N4D4K	150.0	1.06	3295.09	37.8	0.56	820.42
4N2D3K	150.0	5.01	311.78	103.2	3.66	262.33
4N2D4K	150.0	7.58	1319.14	93.8	5.71	1346.59
4N3D3K	150.0	1.32	317.69	6.6	0.15	16.75

Continued on next page

Clustering	SG $\bar{t}$	SG $\overline{\text{rel. DG}}$	SG $\overline{T_{\text{comp}}}$	BTM $\bar{t}$	BTM $\overline{\text{rel. DG}}$	BTM $\overline{T_{\text{comp}}}$
Mean	**54.48**	**1.81**	**483.22**			
4N3D4K	150.0	1.55	2786.47	65.8	0.53	2046.2
4N4D3K	150.0	1.57	441.69	35.0	0.42	122.26
4N4D4K	150.0	1.7	2593.41	7.2	0.14	118.24

Clustering	QNDA $\bar{t}$	QNDA $\overline{\text{rel. DG}}$	QNDA $\overline{T_{\text{comp}}}$
Mean	**54.48**	**1.81**	**483.22**
2N2D3K	63.2	1.82	92.57
2N2D4K	62.2	0.73	345.59
2N3D3K	62.2	1.58	150.11
2N3D4K	5.0	0.06	34.76
2N4D3K	32.8	0.12	97.89
2N4D4K	4.8	0.08	47.19
3N2D3K	121.0	6.21	412.26
3N2D4K	64.2	3.98	747.3
3N3D3K	64.0	1.71	151.79
3N3D4K	63.8	0.28	858.76
3N4D3K	35.0	1.36	111.44
3N4D4K	37.2	0.46	731.33
4N2D3K	94.6	3.61	249.28
4N2D4K	93.8	5.68	1281.33
4N3D3K	9.4	0.17	22.76
4N3D4K	66.0	0.49	1901.69
4N4D3K	37.4	0.41	129.74
4N4D4K	9.8	0.17	178.24