EURO Journal on Computational Optimization 11 (2023) 100079

Contents lists available at ScienceDirect

EURO Journal on Computational Optimization

journal homepage: www.elsevier.com/locate/ejco

Laplacian-based semi-Supervised learning in multilayer hypergraphs by coordinate descent

Sara Venturini [a], Andrea Cristofari [b], Francesco Rinaldi [a,*], Francesco Tudisco [c]

[a] Department of Mathematics "Tullio Levi-Civita", University of Padua, Via Trieste, 63, Padua 35121, Italy
[b] Department of Civil Engineering and Computer Science Engineering, University of Rome "Tor Vergata", Via del Politecnico, 1, Rome 00133, Italy
[c] School of Mathematics, Gran Sasso Science Institute, L'Aquila 67100, Italy

ARTICLE INFO

Keywords:
Semi-supervised learning
Coordinate methods
Multilayer hypergraphs

ABSTRACT

Graph Semi-Supervised learning is an important data analysis tool, where given a graph and a set of labeled nodes, the aim is to infer the labels to the remaining unlabeled nodes. In this paper, we start by considering an optimization-based formulation of the problem for an undirected graph, and then we extend this formulation to multilayer hypergraphs. We solve the problem using different coordinate descent approaches and compare the results with the ones obtained by the classic gradient descent method. Experiments on synthetic and real-world datasets show the potential of using coordinate descent methods with suitable selection rules.

1. Introduction

Consider a finite, weighted and undirected graph $G = (V, E, w)$, with node set V, edge set $E \subseteq V \times V$ and edge-weight function w such that $w(e) = w(uv) > 0$ if $e = (u, v) \in E$ and 0 otherwise. Suppose each node $u \in V$ can be assigned to one of m classes, or labels, $C_1, \cdots, C_m$. In graph-based Semi-Supervised Learning (SSL), given a graph G and an observation set of labeled nodes $O \subset V$ whose vertices $u \in O$ are pre-assigned to some label $y_u \in \{C_1, \cdots, C_m\}$, the aim is to infer the labels of the remaining unlabeled nodes in $V \setminus O$, using the information encoded by the graph [11,60,61].

Extending labels is a-priori an ill-posed problem since there are infinitely many solutions. Therefore, a common approach is to proceed by making the so-called semi-supervised smoothness assumption. This assumption requires that good labeling functions $z_j : V \to \mathbb{R}_+$ for the j-th class, whose entries $z_{u,j}$ quantify the likelihood that $u \in V \setminus O$ belongs to C_j, should be smooth in densely connected regions of the graph. Assuming that the edges of the graph represent some form of similarity between pairs of nodes, this smoothness assumption corresponds to assuming that similar nodes are likely to have similar labels.

Consider the following ℓ_2-based Laplacian regularizer [79]

$$r_2(z) = \frac{1}{2} \sum_{(u,v) \in E} w(uv)(z_u - z_v)^2 . \tag{1}$$

Minimizing $r_2(z)$ subject to either hard label constraints, $z_u = y_u$ for $u \in O$, or a soft penalty constraint like the mean squared error $\sum_u (y_u - z_u)^2$, with respect to the known labels y, is a successful way to enforce smoothness with respect to the edges. In both cases, the

* Corresponding author.
E-mail addresses: sara.venturini@math.unipd.it (S. Venturini), andrea.cristofari@uniroma2.it (A. Cristofari), rinaldi@math.unipd.it (F. Rinaldi), francesco.tudisco@gssi.it (F. Tudisco).

https://doi.org/10.1016/j.ejco.2023.100079

Available online 29 September 2023

resulting objective function is strictly convex and hence the corresponding minimization problem has a unique optimal solution.

Even if the ℓ_2-based Laplacian regularizer is very popular and effective in many situations, it has been proved that it can yield degenerate solutions in the presence of very few input labels in O, because the learned function z becomes nearly constant on the whole graph, with sharp spikes near the labeled data O [22,41]. Therefore, several alternative formulations have been proposed [35,78], including approaches based on total variation [14,30] and the class of p-Laplacian based regularizers [22], more in general, defined as

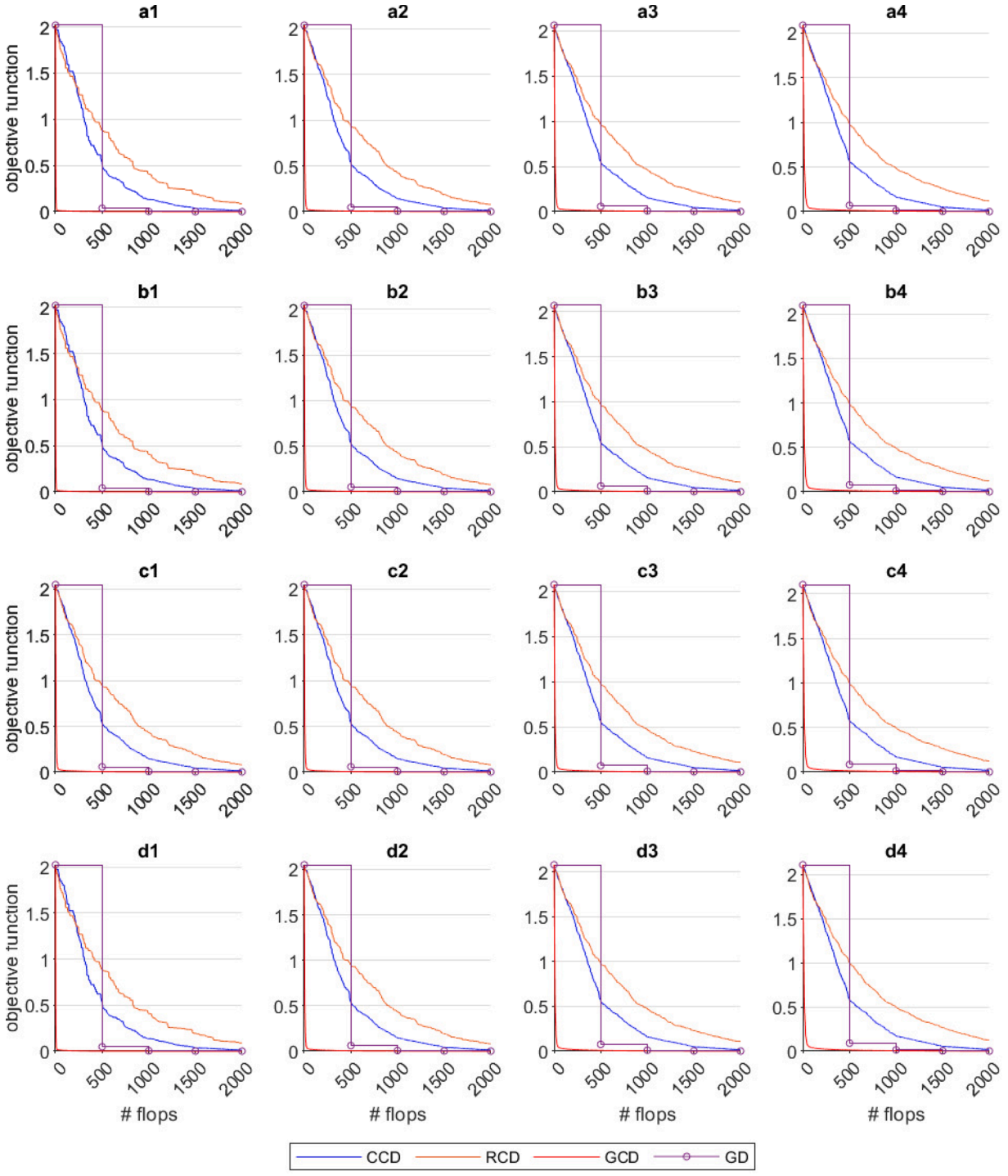

Fig. 1. Average values of the objective function over 5 random networks sampled from SBM for $p = 2$, $p_{in} = 0.2$, $\frac{p_{in}}{p_{out}} \in \{2, 2.5, 3, 3.5\}$ varies in the rows and $perc \in [3\%, 6\%, 9\%, 12\%]$ varies in the columns.

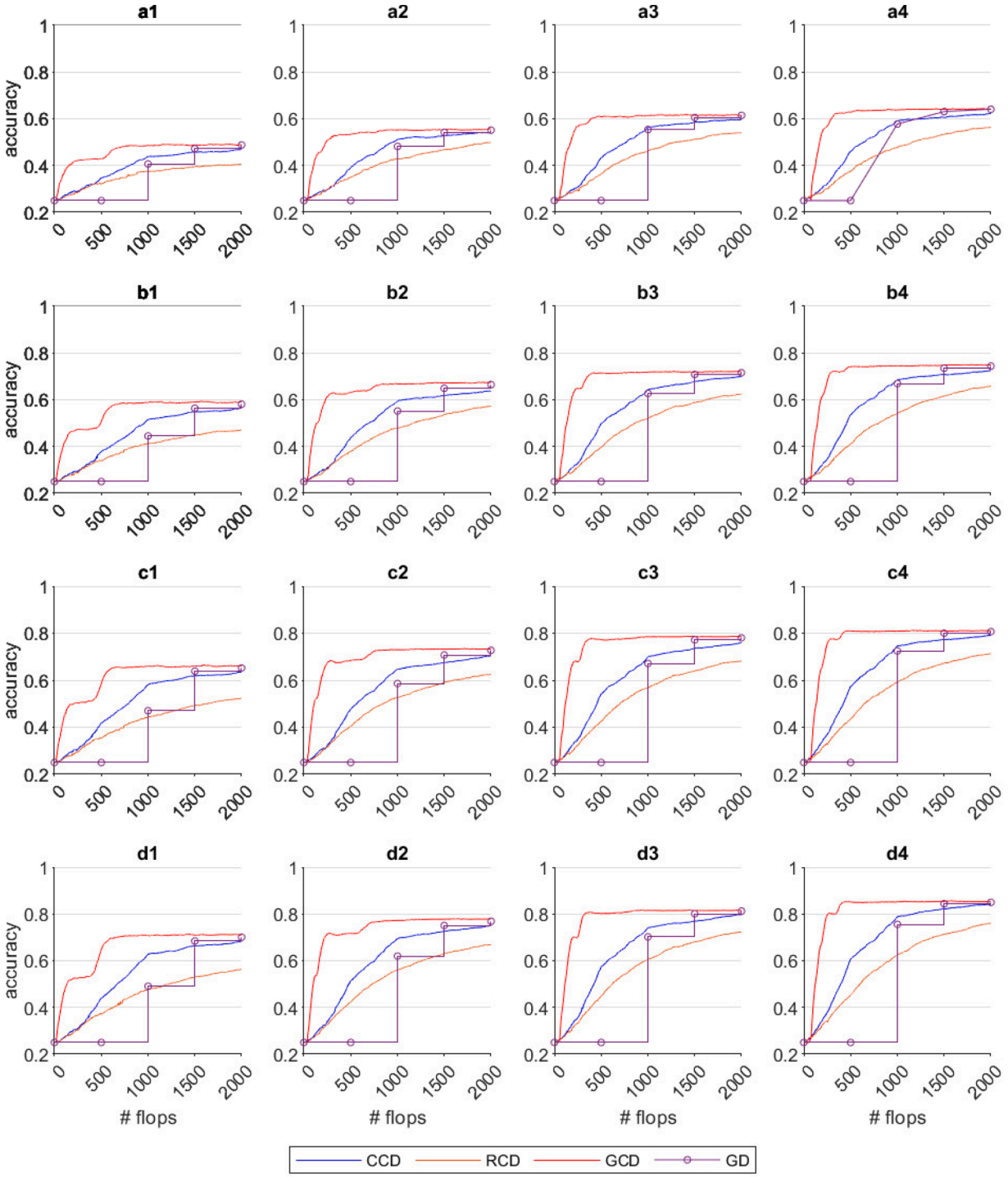

Fig. 2. Average values of the accuracy over 5 random networks sampled from SBM for $p = 2$, $p_{in} = 0.2$, $\frac{p_{in}}{p_{out}} \in \{2, 2.5, 3, 3.5\}$ varies in the rows and $perc \in [3\%, 6\%, 9\%, 12\%]$ varies in the columns.

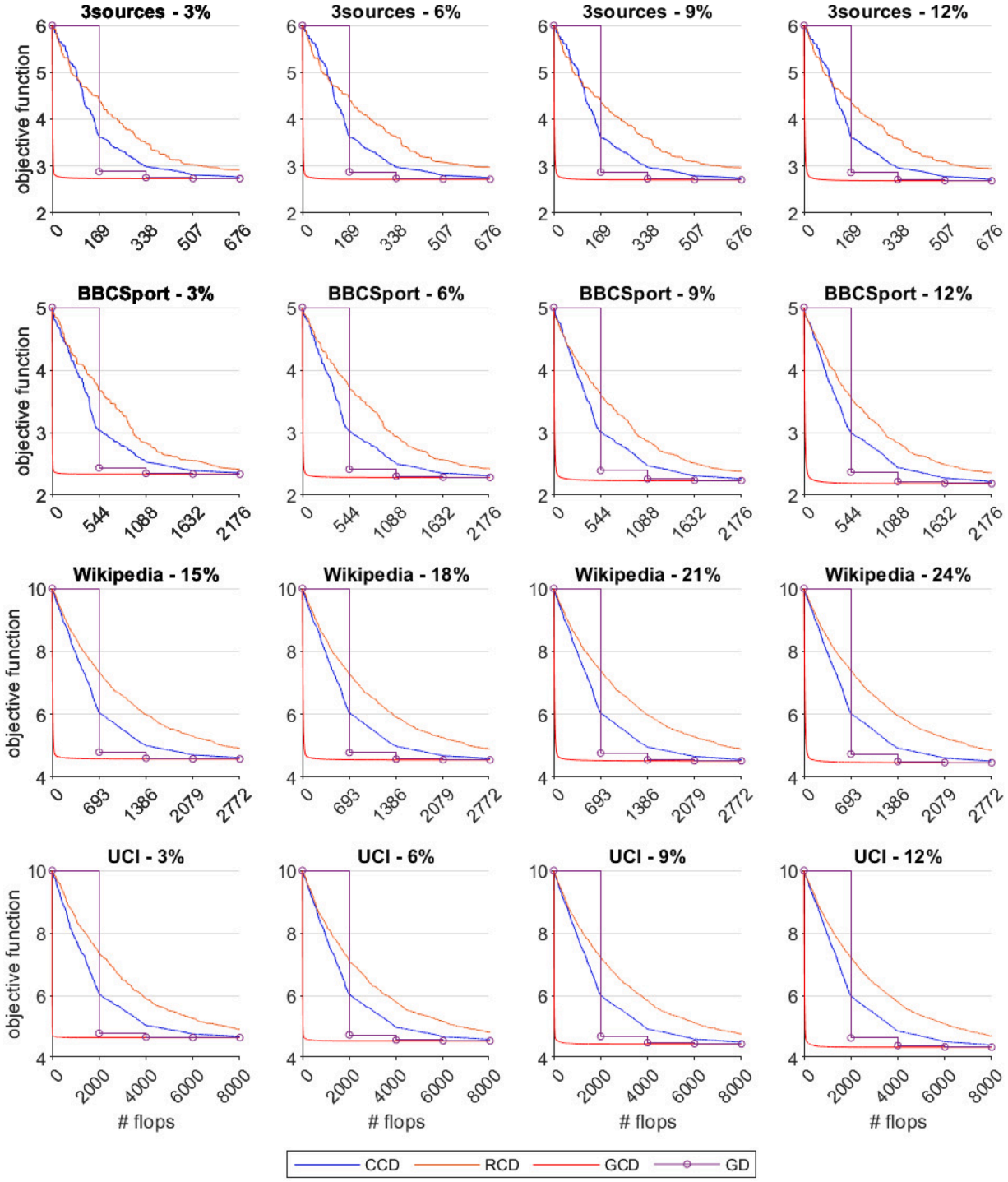

Fig. 3. Average values of the objective function over 5 sampling of know labels, referring to 4 multilayer real-world datasets (3sources, BBCSport, Wikipedia, UCI) with quadratic regularizer. $perc \in [3\%, 6\%, 9\%, 12\%]$ (resp. $perc \in [15\%, 18\%, 21\%, 24\%]$ for Wikipedia) varies in the columns.

$$r_p(z) = \frac{1}{p} \sum_{(u,v)\in E} w(uv)|z_u - z_v|^p \,. \tag{2}$$

Note that this objective function is still strictly convex. Moreover, r_p discourages the solution from developing sharp spikes for $p > 2$, giving a heavier penalty on large gradients $|z_u - z_v|$. Choosing instead $1 \leq p < 2$ encourages the gradient to be sparse. Furthermore, when $p \to 1$, the resulting objective function is directly connected with graph cuts and modular clusters [8,66,68]. Many works studied the behaviour of r_p as p varies, mainly for graphs generated by the geometric random graph model [9,22,24,59,65].

In this paper, we want to investigate the effectiveness of these types of Laplacian regularizers for the task of graph semi-supervised learning, but taking into consideration also higher-order interactions. A variety of complex systems has been successfully described as networks whose interacting pairs of nodes are connected by links. However, in real-world applications, we need to describe interactions in more detailed and varied ways [2,7]. On the one hand, we have simplicial complexes or hypergraphs, which are the natural candidates to describe collective actions of groups of nodes [12,32,69,74,75]. On the other hand, we have multilayer networks, i.e., networks that are coupled to each other through different layers, all of them representing different type of relationships between the nodes [1,23,28,33,40,45,64,77]. Evidence shows that each of those tools can improve modeling capacities with respect to standard graphs. Multilayer hypergraphs arise naturally in diverse applications such as science of science (e.g., nodes represent authors and in one layer a group of authors is an hyperedge if they wrote a paper together, while, in another layer, pairs of nodes are connected if they cite each other), protein networks (e.g., nodes are proteins and they can be connected in pairs or in groups using multiple complementary genomic data which are the different layers), social networks (e.g., nodes are users and they can interact in groups using

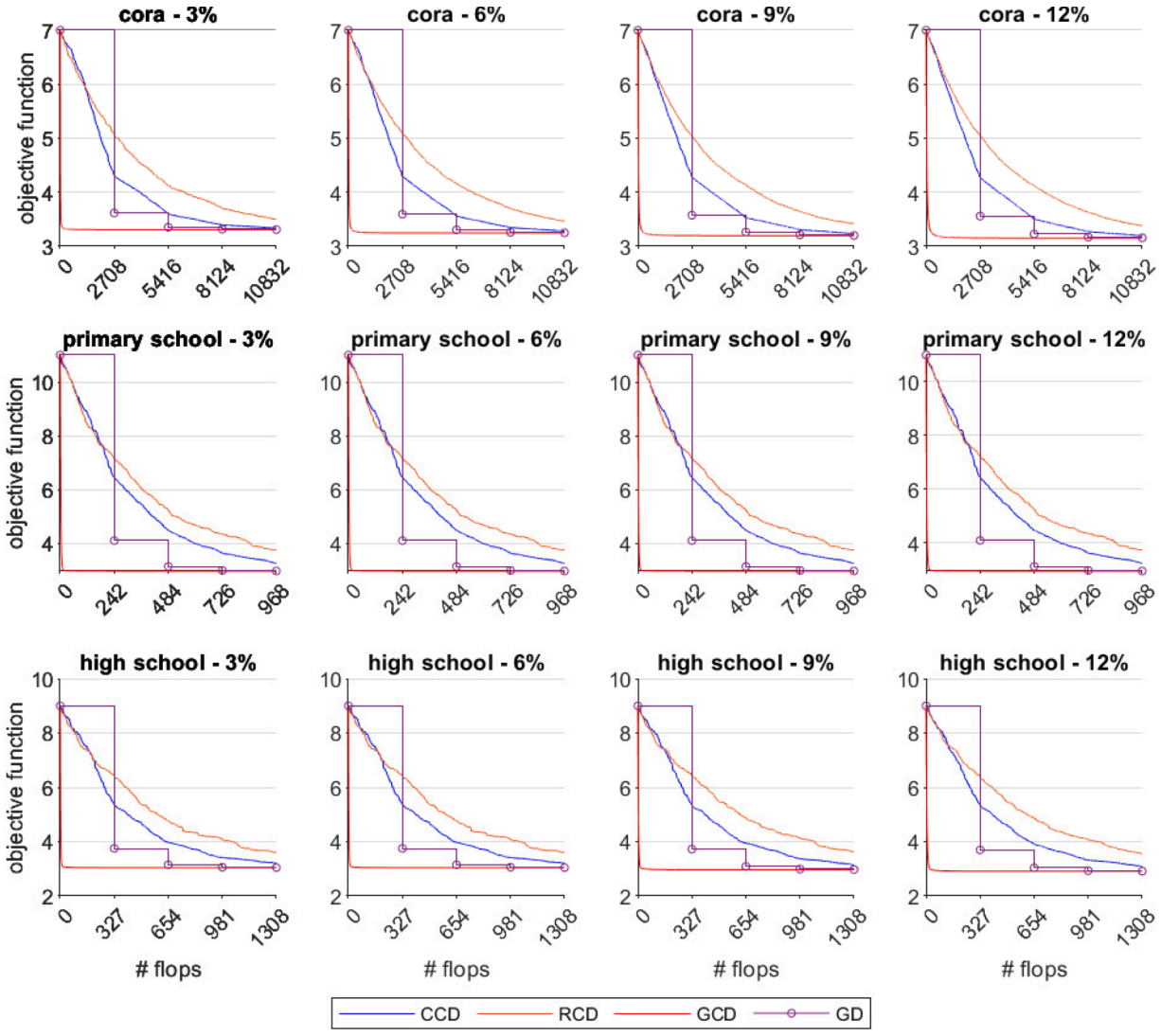

Fig. 4. Average values of the objective function over 5 sampling of know labels, referring to 1 multilayer real-world dataset (cora) and 2 real-world hypergraphs (primary school and high school) with quadratic regularizer. *perc* $\in [3\%, 6\%, 9\%, 12\%]$ varies in the columns.

different platforms). Here, we focus on multiplex hypergraphs, modeled by a sequence of hypergraphs (the layers) with a common set of nodes and no hyperedges between nodes of different layers. Moreover, with the terminology introduced in [39] in the context of multilayer networks, our aim is to find a set of communities that is *total* (i.e., every node belongs to at least one community), *node-disjoint* (i.e., no node belongs to more than one cluster on a single layer), and *pillar* (i.e., each node belongs to the same community across the layers).

However, relatively few studies have considered both multilayer and higher-order structures in complex networks so far [72]. This

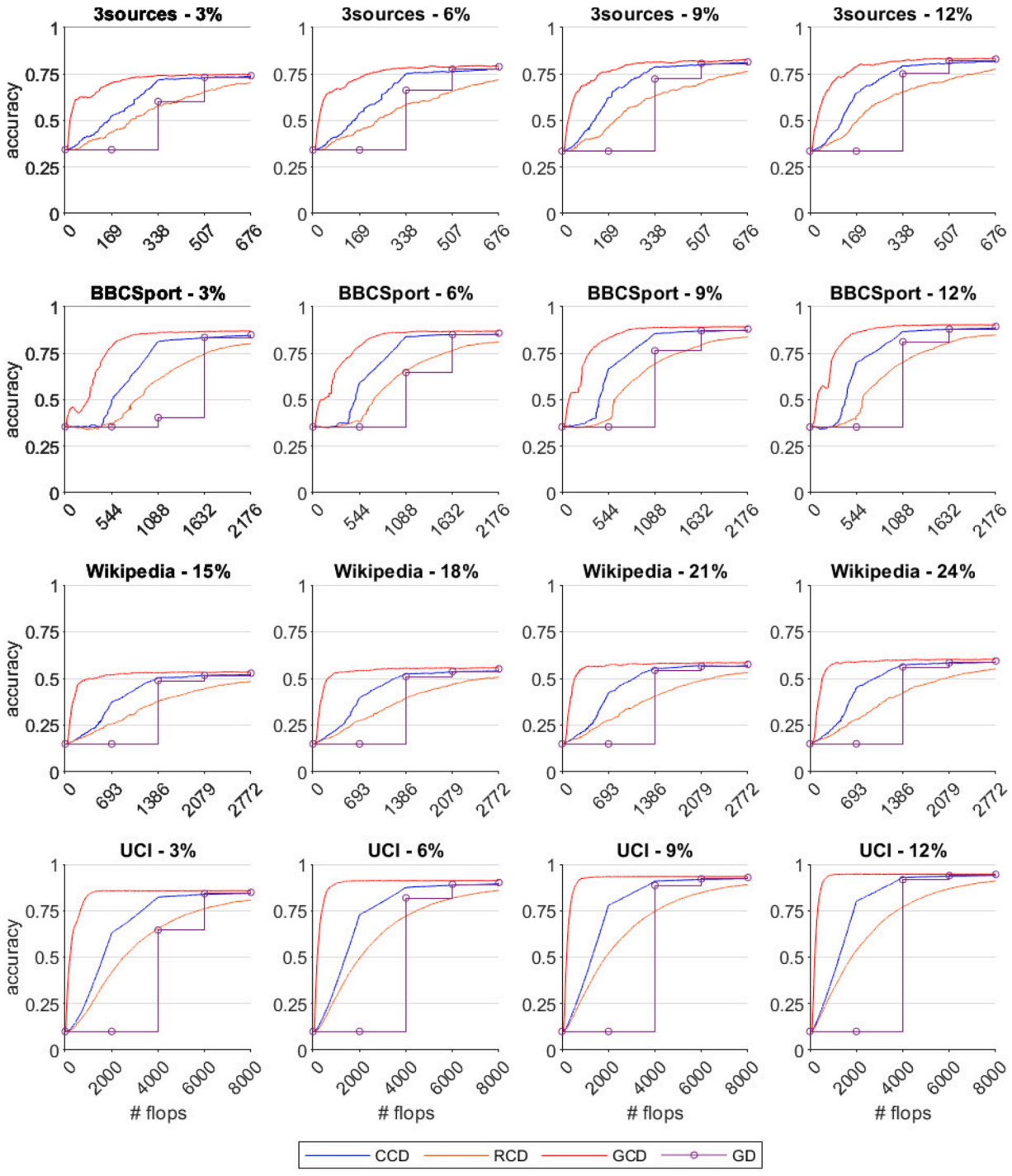

Fig. 5. Average values of the accuracy over 5 sampling of know labels, referring to 4 multilayer real-world datasets (3sources, BBCSport, Wikipedia, UCI) with quadratic regularizer. *perc* $\in [3\%, 6\%, 9\%, 12\%]$ (resp. *perc* $\in [15\%, 18\%, 21\%, 24\%]$ for Wikipedia) varies in the columns.

is mainly due to the fact that getting a good solution for models with such a complex structure comes at a much higher computational cost. In this work, we hence take a first step in the study of semi-supervised learning over multilayer hypergraphs, trying to deal with this additional complexity.

We solve the problem using different coordinate descent approaches and compare the results with the ones obtained by classic first-order approaches, like, e.g., gradient descent/label spreading. Even though coordinate descent approaches were used in the literature to deal with other semi-supervised learning problems [19,21], the analysis reported in this paper represents, to the best of our knowledge, the first attempt to give a thorough analysis of those methods for semi-supervised learning in multilayer hypergraphs.

The rest of the paper is organized as follows. In Section 2, we introduce the graph semi-supervised problem and the formulation for multilayer hypergraphs. In Section 3, we briefly review the block coordinate descent approaches. In Section 4, we report the results of experiments on synthetic and real-world datasets. In Section 5, we draw some conclusions. Finally, in Appendix A, we report some computations needed to apply the methods to our specific problem, pointing out the differences in the special case $p = 2$.

2. Problem statement

In this section, we formalize the notation and formulate the problem under analysis. Consider first an undirected and weighted graph $G = (V, E, w)$ with node set V and edge set E. Let $A = (A_{uv})_{u,v \in V}$ be the adjacency matrix of G, with weights $A_{uv} = w(e) > 0$ for $e = (u, v) \in E$, measuring the strength of the tie between nodes u and v, and $A_{uv} = 0$ if $(u,v) \notin E$. We assume that V can be partitioned into m classes $C_1, \ldots, C_m$ and that, only for a few nodes in $O \subset V$, it is known the class C_j to which they belong. The problem consists in

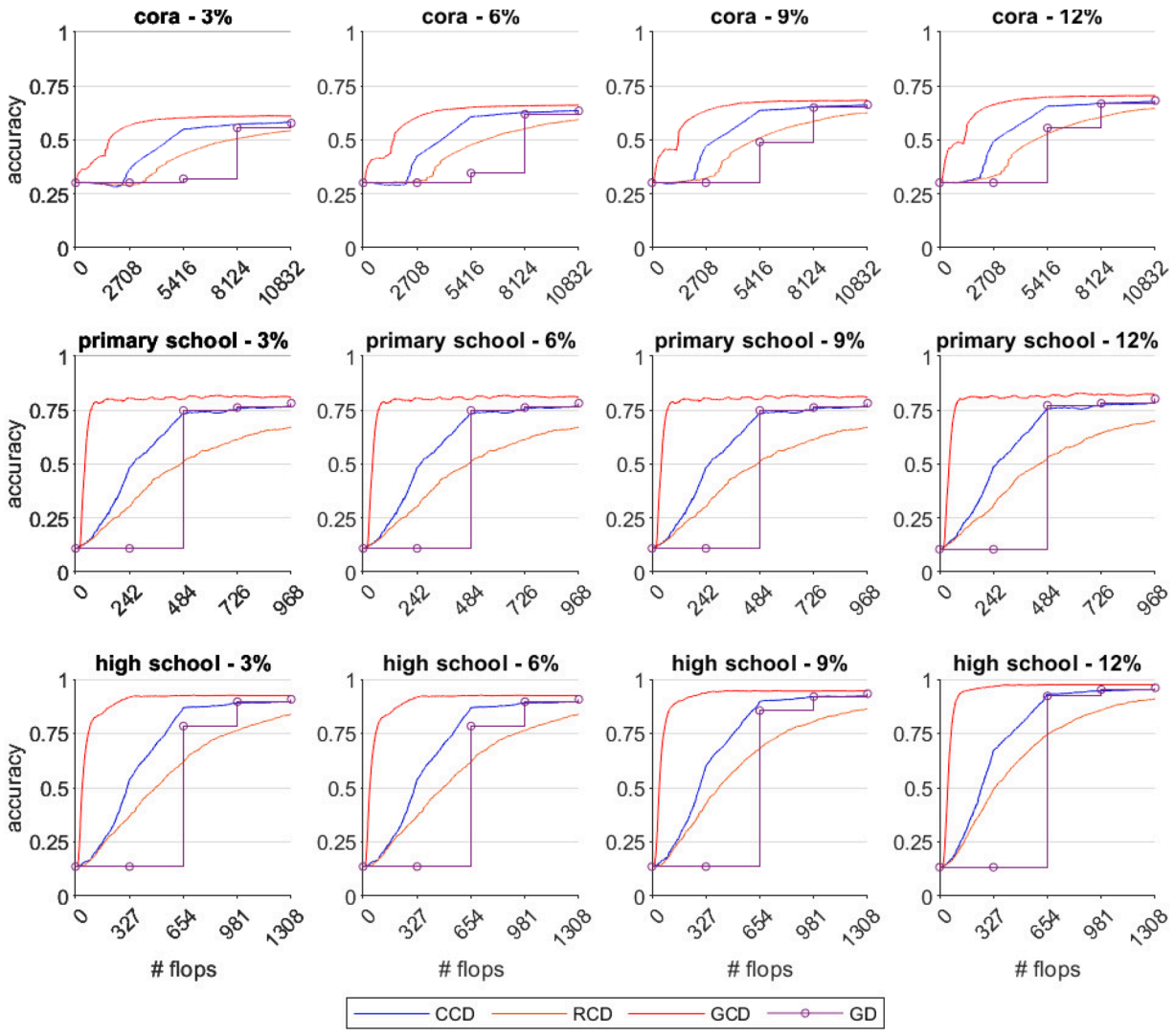

Fig. 6. Average values of the accuracy over 5 sampling of know labels, referring to 1 multilayer real-world dataset (cora) and 2 real-world hypergraphs (primary school and high school) with quadratic regularizer. *perc* $\in [3\%, 6\%, 9\%, 12\%]$ varies in the columns.

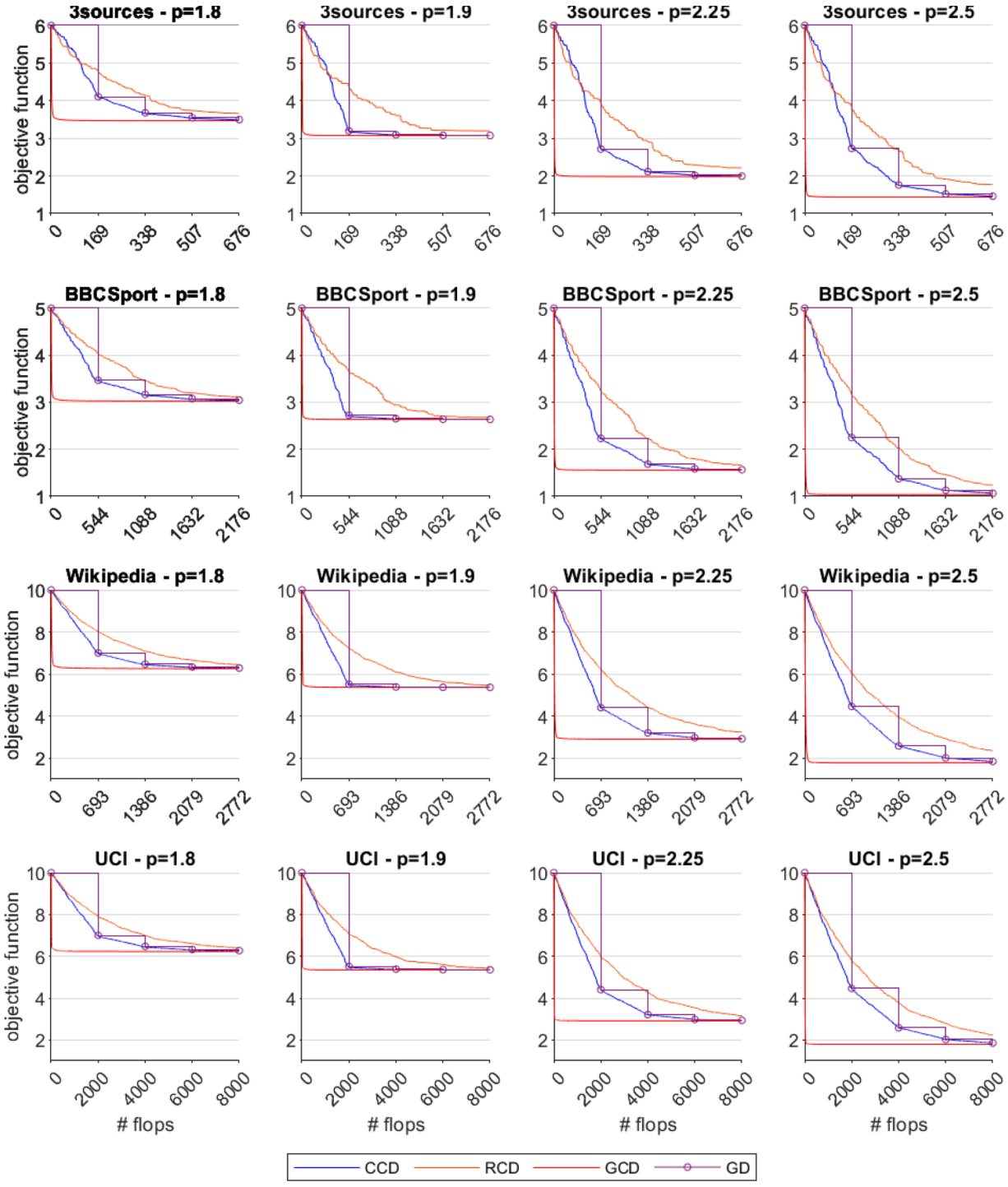

Fig. 7. Average values of the objective function over 5 sampling of know labels, referring to 4 multilayer real-world datasets (3sources, BBCSport, Wikipedia, UCI) with *perc* = 6% (resp. *perc* = 18% for Wikipedia). $p \in [1.8, 1.9, 2.25, 2.5]$ in the regularization term varies in the columns.

assigning the remaining nodes to a class.

Here, we review the approach based on the p-Laplacian regularization and the corresponding optimization problem. Define the $(|V| \times m)$-dimensional matrix of the input labels Y, such that

$$Y_{u,j} = \begin{cases} \dfrac{1}{|C_j \cap O|} & \text{if node } u \in O \text{ belongs to the class } C_j, \\ 0 & \text{otherwise,} \end{cases}$$

where $|C_j \cap O|$ is the cardinality of the known class C_j, i.e., the number of nodes that are initially known to belong to C_j. Now, let y^j be the jth column of Y and, for all $u \in V$, let δ_u be the weighted degree of u, that is, $\delta_u = \sum_{v \in V} A_{uv}$. The Laplacian regularized SSL problem boils down to the following minimization problem for all classes $j \in \{1, \ldots, m\}$:

$$\min_{z \in \mathbb{R}^{|V|}} \| z - y^j \|^2 + \lambda \sum_{u,v \in V} A_{uv} \left| \frac{z_u}{\sqrt{\delta_u}} - \frac{z_v}{\sqrt{\delta_v}} \right|^p, \tag{3}$$

with given $p \geq 1$ and regularization parameter $\lambda \geq 0$. Equivalently, as the minimization problems above are independent for $j \in \{1, \ldots, m\}$, we can simultaneously optimize their sum, which can be written in compact matrix notation as

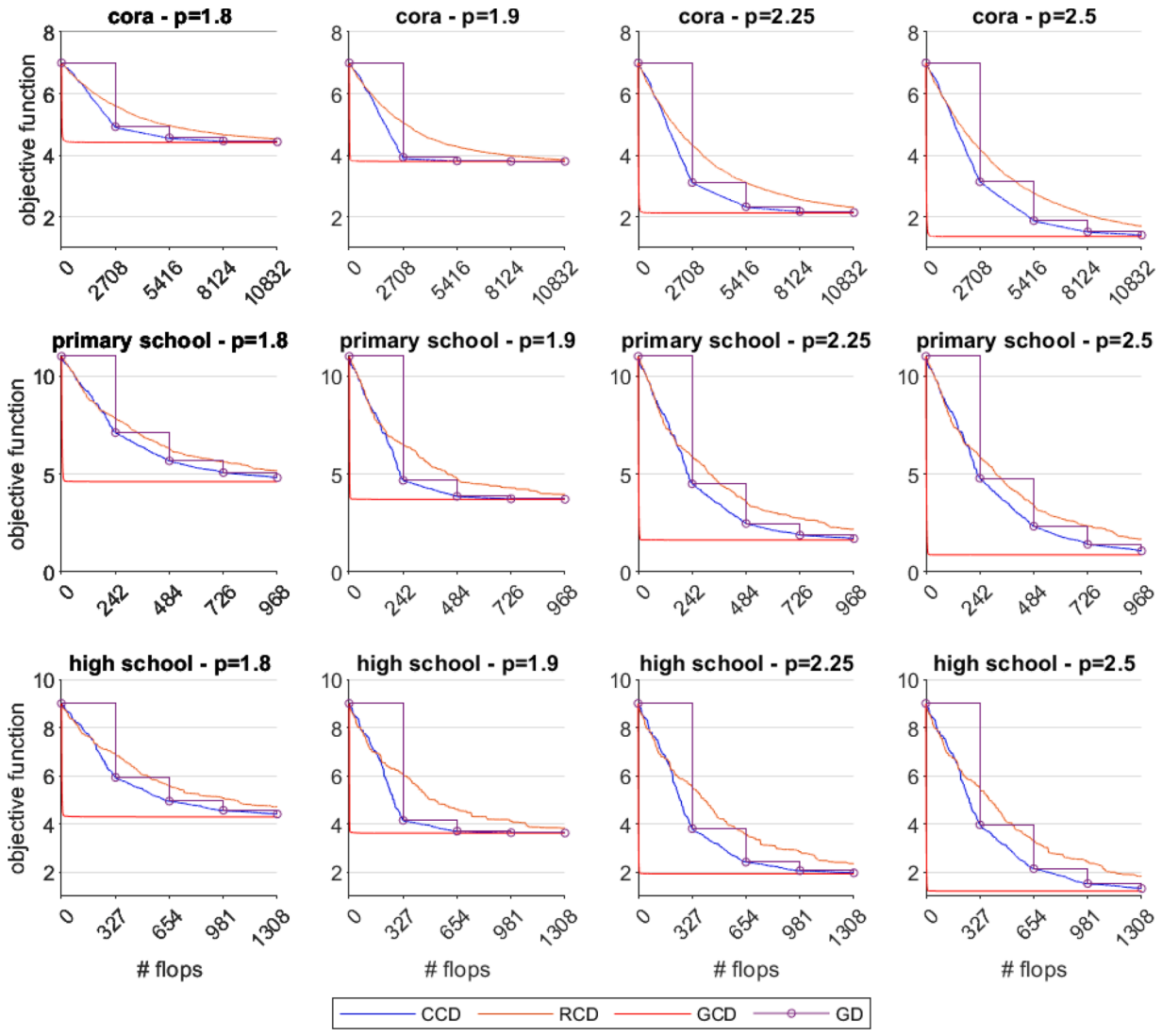

Fig. 8. Average values of the objective function over 5 sampling of know labels, referring to 1 multilayer real-world dataset (cora) and 2 real-world hypergraphs (primary school and high school) with $perc = 6\%$. $p \in [1.8, 1.9, 2.25, 2.5]$ in the regularization term varies in the columns.

$$\min_{Z\in\mathbb{R}^{|V|\times m}} \| Z - Y \|^2_{(2)} + \lambda \| W^{1/p} B D^{-1/2} Z \|^p_{(p)}, \tag{4}$$

where $\| M \|_{(p)}$ denotes the entry-wise ℓ^p norm of the matrix M, D is the $|V| \times |V|$ diagonal matrix of the graph degrees

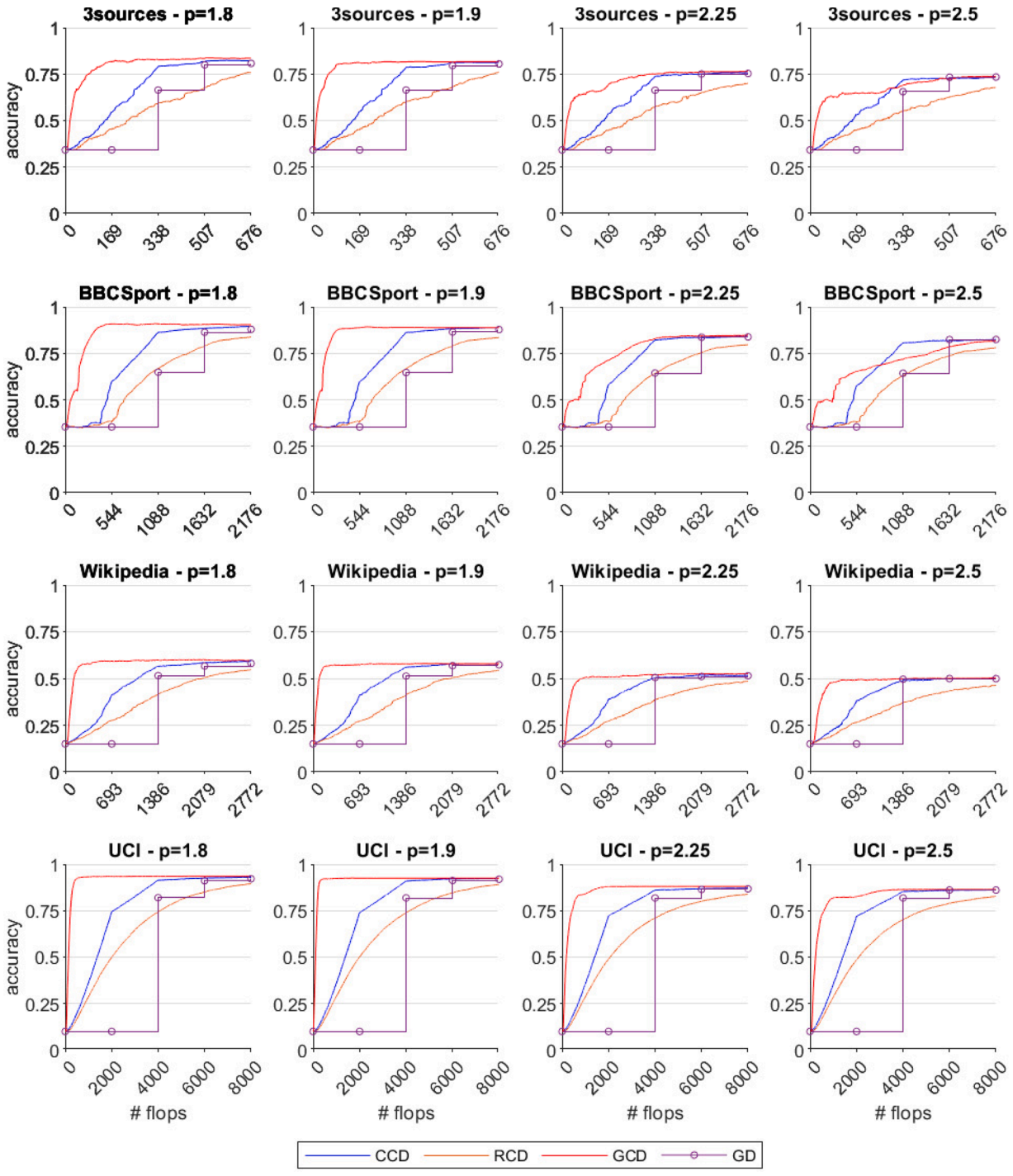

Fig. 9. Average values of the accuracy over 5 sampling of know labels, referring to 4 multilayer real-world datasets (3sources, BBCSport, Wikipedia, UCI) with $perc = 6\%$ (resp. $perc = 18\%$ for Wikipedia). $p \in [1.8, 1.9, 2.25, 2.5]$ in the regularization term varies in the columns.

$$D = \begin{bmatrix} \delta_1 & 0 & \cdots & 0 \\ 0 & \delta_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \delta_{|V|} \end{bmatrix}$$

B is the $|E| \times |V|$ (signed) incidence matrix of the graph, which for any chosen orientation of the edges is entrywise defined as

$$B_{e,u} = \begin{cases} 1 & \text{if node } u \text{ is the source of edge } e, \\ -1 & \text{if node } u \text{ is the tip of edge } e, \\ 0 & \text{otherwise,} \end{cases}$$

and W is the diagonal $|E| \times |E|$ matrix of the edge weights $W_{e,e} = w(e)$. Note that, even though we are dealing with undirected graphs, B requires fixing an orientation for the edges of G. However, all the arguments presented here are independent of the chosen orientation. For $p = 2$, a direct computation shows that the optimal solution Z^* of the above problem is entrywise nonnegative. The same property carries over to any $p \geq 1$, as one can interpret the minimizer of (4) as the smallest solution of a p-Laplacian eigenvalue equation on G with boundary conditions, see e.g. [18]. Thus, we can interpret the entry $Z^*_{u,j} \geq 0$ as a score that quantifies how likely it is for the node $u \in V$ to belong to the class C_j and we then assign each node $u \in V$ to the class $j \in \text{Argmax}_{r=1,\ldots,m} Z^*_{u,r}$.

Now, we want to extend the formulation (4) to the case where rather than a graph G, we have a multilayer hypergraph H. Specifically, assume that we have L layers $H_1, \ldots, H_L$, where $H_\ell(V, E_\ell)$ is the hypergraph forming the ℓth layer and E_ℓ is a hyperedge set,

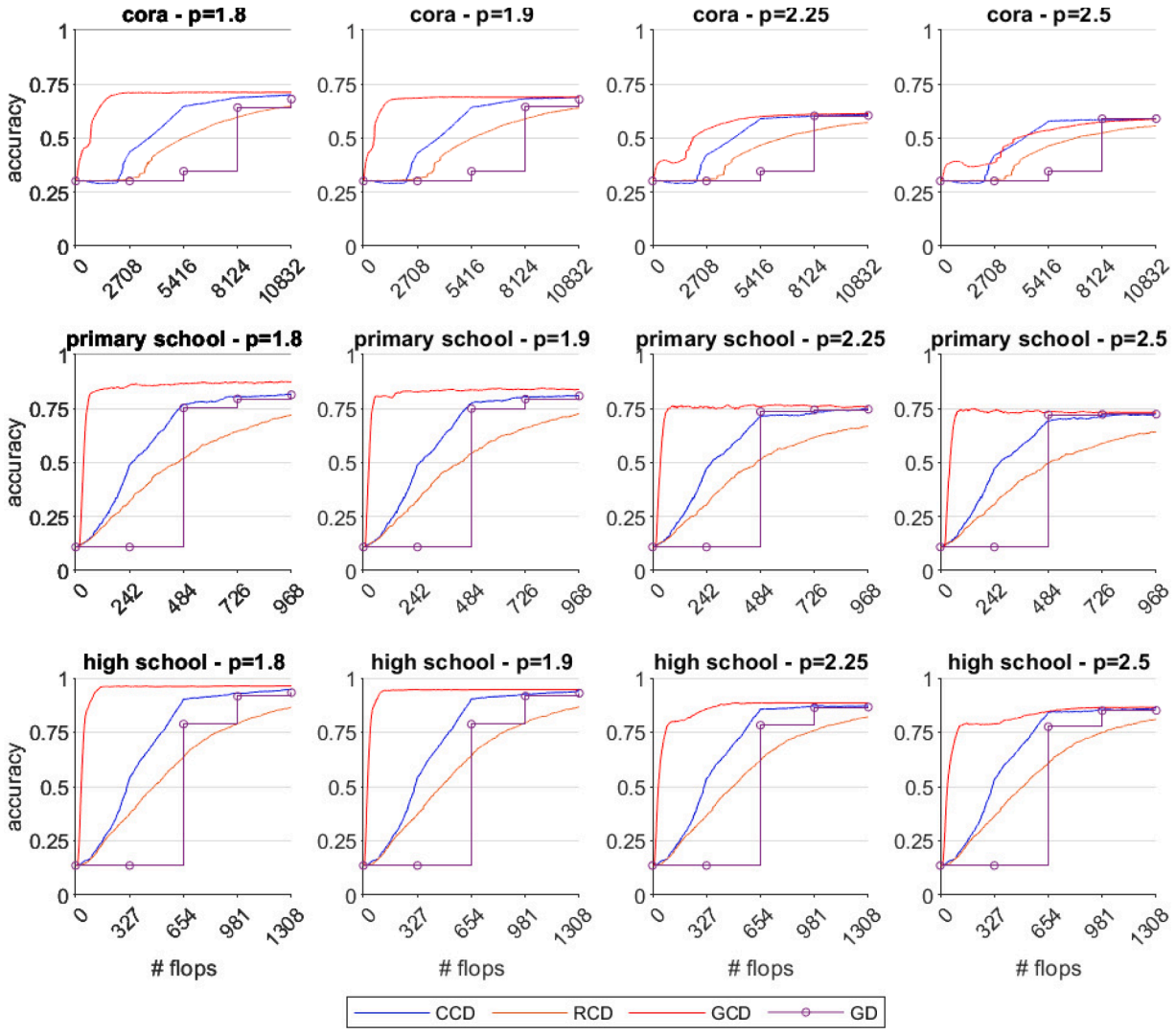

Fig. 10. Average values of the accuracy over 5 sampling of know labels, referring to 1 multilayer real-world dataset (cora) and 2 real-world hypergraphs (primary school and high school) with *perc* = 6%. $p \in [1.8, 1.9, 2.25, 2.5]$ in the regularization term varies in the columns.

that is, E_ℓ contains interactions of order greater than 2. In other words, each $e \in E_\ell$ is a set of arbitrary many nodes, weighted by $w_\ell(e) > 0$. The topological information of a hypergraph H_ℓ can be all included in the (signless) incidence matrix $K_\ell \in \mathbb{R}^{|E_\ell|\times|V|}$, defined as $(K_\ell)_{e,u} = 1$ if $u \in e$, and $(K_\ell)_{e,u} = 0$ if $u \notin e$, for all $u \in V$ and $e \in E_\ell$, see e.g. [2,50,76]. Using K_ℓ, we can represent each $H_\ell(V, E_\ell)$ via a clique-expanded graph $G(H_\ell)$, which corresponds to the adjacency matrix

$$A_\ell = K_\ell^T W_\ell K_\ell - D_\ell,$$

with W_ℓ being the $|E_\ell| \times |E_\ell|$ diagonal matrix of the relative hyperedge weights, defined as

$$(W_\ell)_{e,e} = \frac{w_\ell(e)}{|e|} > 0,$$

and D_ℓ being the diagonal matrix of the node degrees of the hypergraph H_ℓ, defined as

$$(D_\ell)_{u,u} = (\delta_\ell)_u = \sum_{e\in E} w_\ell(e)|e|^{-1}(K_\ell)_{e,u} = \left(K_\ell^T W_\ell K_\ell\right)_{u,u}.$$

Note that the edge (u, v) is in the resulting clique-expanded graph $G(H_\ell)$ if and only if $u \neq v$ and there exists at least one hyperedge in E_ℓ such that both $u \in E_\ell$ and $v \in E_\ell$. In that case, the weight of the edge (u, v) in $G(H_\ell)$ is

$$(A_\ell)_{u,v} = \sum_{e:u,v\in e} \frac{w_\ell(e)}{|e|}$$

and we have $(\delta_\ell)_u = \sum_{v\in V}(A_\ell)_{u,v}$. Therefore, proceeding as before, we can define B_ℓ as the signed incidence matrix of $G(H_\ell)$ and we can sum the corresponding regularization terms across all the layers, obtaining the following formulation:

$$\begin{aligned} &\min_{Z\in\mathbb{R}^{|V|\times m}} \quad \vartheta(Z) := f(Z) + r_p(Z) \\ &\text{where} \quad f(Z) = \| Z - Y \|_{(2)}^2, \quad r_p(Z) = \sum_{\ell=1}^{L} \lambda_\ell \| W_\ell^{1/2} B_\ell D_\ell^{-1/2} Z \|_{(p)}^p, \end{aligned} \tag{5}$$

where $\lambda_1, \ldots, \lambda_L \geq 0$ are regularization parameters. Note that, if H is a standard graph, i.e., if $|e| = 2$ for all edges and $L = 1$, then (5) boils down to (4), up to the constant term $1/|e| = 1/2$. Note moreover that, as in the graph case, we can equivalently write the objective function $\vartheta(Z)$ as $\sum_j \vartheta_j(z^j)$, where z^j is the j-th column of Z, and

$$\vartheta_j(z) = \| z - y^j \|_2^2 + \sum_{\ell=1}^{L} \lambda_\ell \sum_{e\in E_\ell} \frac{w_\ell(e)}{|e|} \sum_{u,v\in e} \left| \frac{z_u}{\sqrt{(\delta_\ell)_u}} - \frac{z_v}{\sqrt{(\delta_\ell)_v}} \right|^p.$$

The above expression shows that the regularizers ϑ_j enforce a form of higher-order smoothness assumption in the solution across all the nodes of each layer's hyperedge by imposing the minimizer Z^* to have similar values on pairs of nodes in the same hyperedge. This immediately justifies the choice of the objective function (5) for SSL on multilayer hypergraphs. Also note that, as in the graph setting, the optimal solution Z^* to (5) has to be entrywise nonnegative and thus, once Z^* is computed, we can assign each node $u \in V$ to the class $j \in \text{Argmax}_{r=1,\ldots,m} Z^*_{u,r}$.

3. Block coordinate descent approaches

When dealing with large-scale optimization problems, such as those arising in semi-supervised learning problems on real-world multilayer hypergraphs, traditional methods may be impractical and block coordinate descent methods represent a valid tool to achieve high efficiency. At every iteration of a block coordinate descent method, a working set of a few variables is suitably selected and properly updated, while keeping the remaining variables fixed. The general scheme for a block coordinate descent method to minimize an objective function $f(z)$ is reported in Algorithm 1.

In the literature, many block coordinate descent methods were proposed for both unconstrained and constrained problems, differing from each other in the computation of W^k and s^k (see, e.g., [73] and the references therein). As for the computation of the working set W^k, a possible choice is to use a *cyclic rule*, also known as *Gauss-Seidel rule* [4]. It consists in partitioning the variables into a number of blocks and selecting each of them in a cyclic fashion. This approach can be generalized to the *essentially cyclic rule* or *almost cyclic rule* [38], requiring that each block of variables must be selected at least once within a prefixed number of iterations. In unconstrained optimization, blocks can even be made of just one variable and every update (i.e., the computation of s^k) can be carried out by an exact or an inexact minimization [4,5,26,38,58]. These methods have been also extended to constrained settings, possibly requiring blocks being made of more than one variable when the constraints are not separable [4,6,10,15,27,37,52]. A remarkable feature of cyclic based rules is that, at each iteration, only a few components of ∇f must be calculated. This can lead to high efficiency when computing one component of ∇f is much cheaper than computing the whole gradient vector.

A second possibility to choose the working set is to use a *random rule*, that is, W^k can be computed randomly from a given

0: **Given** $z^0 \in \mathbb{R}^n$
1: **For** $k = 0, 1, \ldots$
2: Choose a working set $W^k \subseteq \{1, \ldots, n\}$
3: Compute $s^k \in \mathbb{R}^n$ such that $s_i^k = 0$ for all $i \notin W^k$
4: Set $z^{k+1} = z^k + s^k$
5: **End for**

Algorithm 1. Generic block coordinate descent method.

0: **Given** $Z^0 \in \mathbb{R}^{n\times m}$
1: **For** $k = 0, 1, \ldots$
2: Choose a working set $W^k = W_1^k \times \ldots \times W_m^k \subseteq \{1, \ldots, n\}^m$
3: Compute $S^k \in \mathbb{R}^{n\times m}$ such that $S_{ij}^k = 0$ for all $i \notin W_j^k$
4: Set $Z^{k+1} = Z^k + S^k$
5: **End for**

Algorithm 2. Block coordinate descent method for problem (5) - matrix form.

probability distribution. These algorithms, usually known as *random coordinate descent methods*, show nice convergence properties in expectation for both unconstrained [44,56] and constrained problems [25,42,43,49,53]. Note that random rules, as well as cyclic rules, do not use first-order information to compute the working set, thus still leading to high efficiency when the computation of one component of ∇f is much cheaper than the computation of the whole gradient vector.

Another way to choose the working set W^k is to use a *greedy rule*, also known as *Gauss-Southwell rule*. It consists in selecting, at each iteration, a block containing the variable(s) that most violate a given optimality condition. In the unconstrained case we can choose as working set, for instance, the block corresponding to the largest component of the gradient in absolute value. Also for this rule, exact or inexact minimizations can be carried out to update the variables [16,17,26,38] and extensions to constrained settings were considered in the literature [3,36,48,62,63]. Generally speaking, a greedy rule might make more progress in the objective function, since it uses first (or higher) order information to choose the working set, but might be, in principle, more expensive than cyclic or random selection. However, several recent works show that certain problem structures allow for efficient calculation of this class of rules in practice (see, e.g., [48] and references therein for further details).

In this work, we adapt block coordinate descent methods to solve problem (5), leading to the method reported in Algorithm 2. In particular, we start with a matrix $Z^0 \in \mathbb{R}^{n\times m}$ and, at each iteration k, we choose a working set W_j^k for each class $j \in \{1, \ldots, m\}$. We highlight that problem (5) solves the same problem for the different classes C_j with $j = 1, \ldots, m$ in a matrix form, but each of them is independent and can eventually be solved in parallel.

3.1. Coordinate descent approaches

In this paper, we focus on block coordinate descent approaches that use blocks W_j^k of dimension 1, i.e., $W_j^k = \{i_j^k\}$, with i_j^k being a variable index for class j at iteration k. Then, Z^{k+1} is obtained by moving the variables $Z_{i_j^k j}^k$ along $-\nabla_{i_j^k j}\vartheta(Z^k)$ with a proper stepsize α_j^k. Namely, for any class $j \in \{1, \ldots, m\}$,

$$Z_{hj}^{k+1} = \begin{cases} Z_{hj}^k - \alpha_j^k \nabla_{hj}\vartheta(Z^k) & \text{if } h = i_j^k, \\ Z_{hj}^k & \text{otherwise.} \end{cases} \tag{6}$$

Taking into account the possible choices described in Section 3, we consider the following algorithms:

- **Cyclic Coordinate Descent (CCD).** At every iteration k, a variable index $i^k \in \{1, \ldots, n\}$ is chosen in a cyclic fashion (i.e., by a Gauss-Seidel rule), and then Z^{k+1} is obtained as in (6) by setting $i_j^k = i^k$ for all $j \in \{1, \ldots, m\}$. A random permutation of the variables every n iterations is also used, since it is known that this might lead to better practical performances in several cases (see, e.g., [29,73]).
- **Random Coordinate Descent (RCD).** At every iteration k, a variable index $i^k \in \{1, \ldots, n\}$ is randomly chosen from a uniform distribution, and then Z^{k+1} is obtained as in (6) by setting $i_j^k = i^k$ for all $j \in \{1, \ldots, m\}$.
- **Greedy Coordinate Descent (GCD).** At every iteration k, a variable index $i_j^k \in \{1, \ldots, n\}$ is chosen for every class $j \in \{1, \ldots, m\}$ as

$$i_j^k \in \operatorname{Argmax}_{i=1,\ldots,n} \left|\nabla_{ij}\vartheta(Z^k)\right|$$

(i.e., by a Gauss-Southwell rule), and then Z^{k+1} is obtained as in (6).

The GCD method guarantees good rates when proper conditions are met [48]. Anyway, since at each iteration we need to evaluate the whole gradient and search for the best index in order to choose the block to be used in the update, it might become very expensive when we tackle large-scale semi-supervised learning problems. To practically implement those methods, specific strategies hence need to be implemented. It is important to highlight that, due to the sparsity present in the semi-supervised learning problems we consider, it is possible to implement the basic GCD rule in an efficient way (by, e.g., tracking the gradient element in a max-heap structure, using caching strategies), see, e.g., [48].

Since the practical efficiency of coordinate methods strongly depends on how the algorithm is implemented, we report, in Appendix A, details on the calculations needed to update the gradient of the objective functions at a given iteration.

4. Numerical experiments

First-order methods like, e.g., gradient descent/label spreading are widely used in the context of semi-supervised learning [46,61,67]. This is the reason why we compare the coordinate approaches described in Subsection 3.1, i.e., the Cyclic Coordinate Descent method (CCD), Random Coordinate Descent method (RCD) and Greedy Coordinate Descent method (GCD), with the Gradient Descent (GD) in our experiments. In the first setting, when $p = 2$, the objective function is quadratic and we used a stepsize depending on the coordinatewise Lipschitz constants (see, e.g., [44]). We highlight that while calculating the coordinatewise Lipschitz constants or a good upper bound is pretty straightforward in the considered case for coordinate approaches, the calculation of the global Lipschitz constant might get expensive for GD (especially when dealing with large-scale instances). For the $p \neq 2$ setting, for simplicity we used a stepsize depending on an upper bound of the Lipschitz constants for all the methods (see, e.g., [34,47,48]). The performance of the coordinate descent algorithms might of course be further improved by choosing a more sophisticated coordinate dependent stepsize strategy [51,54,55,57]. In order to show the advantages of using coordinate methods with respect to gradient descent-like approaches, and how efficient those methods are in practice, we performed extensive experiments both on synthetic and real-world datasets.

We report the efficiency plots of the objective function and accuracy of the final partition (evaluated on the subset of unlabeled nodes). In our experiments, we use the number of flops (i.e., one-dimensional moves) as our measure of performance. Therefore, for a graph with N nodes, the GD uses N flops at each iteration (i.e., it changes all the N components of the iterate), while the coordinate methods use just one flop per iteration (i.e., they change just one component at the time). As already highlighted in [48], this measure is far from perfect, especially when considering greedy methods, since it ignores the computational cost of each iteration. However, it gives an implementation- and problem-independent measure. Furthermore, in our case, it is easy to estimate the cost per iteration (which is small when the strategy is suitably implemented). Thus, we will see how a faster-converging method like GCD leads to a substantial performance gain on the considered application.

We fixed the regularization parameters at $\lambda_\ell = 1$ for $\ell = 1, \dots, L$ and we initialized the methods with $Z^0 = 0$. We implemented all the methods using Matlab. We emphasize that the choice of the parameters $\lambda_\ell = 1$ does not affect the performance analysis we carry out in this work and has been made to ensure a fair balance among all layers. In practice, the choice of these parameters may require a non-trivial parameter tuning phase which is typically either model- or data-based, see e.g. [45,64,71]. Code and data to reproduce the experiments are available at the repository https://github.com/saraventurini/Semi-Supervised-Learning-in-Multilayer-Hypergraphs-by-Coordinate-Descent.

4.1. Synthetic datasets

We generated synthetic datasets by means of the Stochastic Block Model (SBM) [31], a generative model for graphs with planted communities depending on suitably chosen parameters p_{in} and p_{out}. Those parameters represent the edge probabilities: given nodes u

Table 1
Aggregated results of the objective function (upper table) and the accuracy (lower table) across the synthetic datasets with $p = 2$ (see Figs. 1 and 2). Using a tolerance *gate*, for each algorithm *flop* indicates the normalized number of flops (mean ± standard deviation) and *fail* indicates the fraction of failures (i.e., stopping criterion not satisfied within the maximum number of iterations, set equal to 4 times the number of nodes). The averages are calculated without considering the failures and, in case of all failures, a hyphen is reported.

	CCD		RCD		GCD		GD	
gate	flop	fail	flop	fail	flop	fail	flop	fail
0.75	0.65±0.07	0.00	0.80±0.06	0.00	0.15±0.04	0.00	2.00±0.00	0.00
0.5	0.66±0.04	0.00	0.88±0.06	0.00	0.02±0.01	0.00	1.00±0.00	0.00
0.25	1.05±0.04	0.00	1.83±0.06	0.25	0.02±0.01	0.00	1.00±0.00	0.00
0.1	1.82±0.05	0.00	3.08±0.16	0.00	0.04±0.02	0.00	1.00±0.00	0.00
0.05	2.44±0.08	0.00	3.81±0.06	0.50	0.05±0.03	0.00	1.00±0.00	0.00
	CCD		RCD		GCD		GD	
gate	flop	fail	flop	fail	flop	fail	flop	fail
0.75	0.65±0.07	0.00	0.80±0.06	0.00	0.15±0.04	0.00	2.00±0.00	0.00
0.5	1.06±0.15	0.00	1.71±0.28	0.00	0.24±0.03	0.00	2.00±0.00	0.00
0.25	1.75±0.13	0.00	3.28±0.29	0.25	0.54±0.27	0.00	2.44±0.51	0.00
0.1	3.07±0.53	0.00	-	1.00	0.82±0.28	0.00	3.00±0.00	0.00
0.05	3.62±0.32	1.38	-	1.00	1.02±0.32	0.00	3.50±0.52	0.00

and v, the probability of observing an edge between them is p_{in} (resp. p_{out}) if u and v belong to the same (resp. different) cluster.

Notice that solving problem (5) on a multilayer hypergraph is equivalent to solving the same problem over a simple graph with an adjacency matrix made by the weighted sum of the adjacency matrices of the clique-expanded graphs of each layer. Therefore, we generated single layer datasets by fixing $p_{in} = 0.2$ and varying the ratio $p_{in}/p_{out} \in \{3.5, 3, 2.5, 2\}$. More precisely, we created networks with 4 communities of 125 nodes each. We tested the methods also considering different percentages *perc* of known labels per community. In particular, we consider $perc \in \{3\%, 6\%, 9\%, 12\%\}$, i.e., respectively $3, 7, 11, 15$ known nodes per community.

We studied the optimization problem (5) fixing $p = 2$. For each value of $(p_{out}, perc)$ we sampled 5 random instances and considered average scores. Results reported in Figs. 1 and 2 respectively show the value of the objective function and the accuracy, in terms of number of flops. In each row, we report the results related to a fixed value of the ratio $p_{in}/p_{out} \in \{2, 2.5, 3, 3.5\}$ (ratio increasing top to bottom), varying the percentage of known labels $perc \in \{3\%, 6\%, 9\%, 12\%\}$ (percentage increasing left to right). In Table 1, we present aggregated results of the objective function and the accuracy across the synthetic datasets (see Figs. 1 and 2). For each method, it is shown average and standard deviation of the number of flops, normalized by the total number of nodes in the network, required to reach a certain level of objective/accuracy. This depends by a *gate*, that is, a convergence tolerance, as in [20]. It is also shown the fraction of failures, i.e., the fraction of problems where a method does not convergence within a number of iterations equal to 4 times the number of nodes. The averages are calculated without considering the failures and, in case of all failures, a hyphen is reported. As we can easily see by taking a look at the plots and the tables, GCD always reaches a good solution in terms of both objective function value and accuracy, with a much lower number of flops than the other methods under consideration. Concerning the other coordinate methods under analysis, they seem to be slower than GD in getting a good objective function, but faster in terms of accuracy. Therefore, if the coordinate selection is properly carried out, a coordinate method might outperform GD in practice.

4.2. Real datasets

We further consider seven real-world datasets frequently used for assessing algorithm performance in graph clustering (information can be found in the GitHub repository) [13,40,70]:

- *3sources*: 169 nodes, 6 communities, 3 layers;
- *BBCSport*: 544 nodes, 5 communities, 2 layers;
- *Wikipedia*: 693 nodes, 10 communities, 2 layers;
- *UCI*: 2000 nodes, 10 communities, 6 layers;
- *cora*: 2708 nodes, 7 communities, 2 layers;
- *primary-school*: 242 nodes, 11 communities, 2.4 mean hyperedge size;
- *high-school*: 327 nodes, 9 communities, 2.3 mean hyperedge size.

The first five datasets in the list are related to multilayer graphs, while the last two are related to single layer hypergraphs.

We tested the methods considering different percentages of known labels per community, sampling them randomly 5 times and showing the average scores. In particular, we suppose to know $perc \in [3\%, 6\%, 9\%, 12\%]$ percentage of nodes per community in all the datasets except for *Wikipedia*, where we considered to know a higher percentage of nodes, $perc \in [15\%, 18\%, 21\%, 24\%]$, to have significant results.

Firstly, we analyze the results corresponding to a quadratic regularization in (5) (fixing $p = 2$). In Figs. 3 and 4, we report the average values of the objective function, and in Figs. 5 and 6, the related accuracy values. In Table 2, we present aggregated results of the objective function and the accuracy, as explain in Section 4.1. We can see that the results match the ones obtained for the synthetic datasets. The Greedy Coordinate Descent method (GCD) indeed reaches a good solution in terms of both objective function and accuracy, with a much lower number of flops than the other methods.

Table 2
Aggregated results of the objective function (upper table) and the accuracy (lower table) across the real datasets with $p = 2$ (see Figs. 3–4 and Figs. 5–6). The table indices are the same as in Table 1.

	CCD		RCD		GCD		GD	
gate	flop	fail	flop	fail	flop	fail	flop	fail
0.75	0.39± 0.06	0.00	0.40±0.02	0.00	0.01±0.00	0.00	1.00±0.00	0.00
0.5	0.74±0.08	0.00	1.06±0.09	0.00	0.01±0.00	0.00	1.00±0.00	0.00
0.25	1.30±0.24	0.00	2.06±0.16	0.00	0.02±0.01	0.00	1.00±0.00	0.00
0.1	2.16±0.40	0.00	3.37±0.36	0.07	0.03±0.01	0.00	1.32±0.48	0.00
0.05	2.90±0.49	0.00	3.49±0.00	0.96	0.04±0.02	0.00	1.61±0.50	0.00
	CCD		RCD		GCD		GD	
gate	flop	fail	flop	fail	flop	fail	flop	fail
0.75	0.69±0.17	0.00	0.97±0.28	0.00	0.13±0.08	0.00	2.11±0.31	0.00
0.5	0.99±0.20	0.00	1.61±0.31	0.00	0.25±0.16	0.00	2.15±0.36	0.00
0.25	1.51±0.26	0.00	2.78±0.49	0.00	0.41±0.24	0.00	2.32±0.48	0.00
0.1	2.15±0.52	0.00	3.21±0.23	0.82	0.67±0.39	0.00	2.71±0.54	0.04
0.05	2.70±0.56	0.29	3.86±0.00	0.96	0.91±0.52	0.00	3.10±0.44	0.25

Table 3
Aggregated results of the objective function (upper table) and the accuracy (lower table) across the real datasets with $p \neq 2$ (see Figs. 7–8 and Figs. 9–10). The table indices are the same as in Table 1.

	CCD		RCD		GCD		GD	
gate	flop	fail	flop	fail	flop	fail	flop	fail
0.75	0.35±0.06	0.00	0.35±0.05	0.00	0.01±0.00	0.00	1.00±0.00	0.00
0.5	0.66±0.09	0.00	0.93±0.14	0.00	0.01±0.00	0.00	1.00±0.00	0.00
0.25	1.06±0.26	0.00	1.83±0.21	0.00	0.01±0.01	0.00	1.43±0.50	0.00
0.1	1.68±0.49	0.00	2.99±0.43	0.00	0.02±0.01	0.00	1.93±0.60	0.00
0.05	2.13±0.72	0.00	3.40±0.39	0.50	0.03±0.01	0.00	2.50±0.29	0.00
	CCD		RCD		GCD		GD	
gate	flop	fail	flop	fail	flop	fail	flop	fail
0.75	0.70±0.16	0.00	0.97±0.28	0.00	0.09±0.03	0.00	2.15±0.36	0.00
0.5	1.01±0.19	0.00	1.64±0.29	0.00	0.23±0.24	0.00	2.15±0.36	0.00
0.25	1.51±0.26	0.00	2.72±0.45	0.00	0.41±0.45	0.00	2.36±0.49	0.00
0.1	1.93±0.26	0.00	3.46±0.46	0.68	0.72±0.76	0.00	2.90±0.57	0.00
0.05	2.39±0.54	0.04	3.90±0.11	0.89	0.94±0.90	0.00	3.05±0.56	0.18

Table 4
Maximum value of accuracy achieved in the real datasets with fixed $perc = 6\%$ (resp. $perc = 18\%$ for Wikipedia) and varing $p \in \{1.8, 1.9, 2, 2.25, 2.5\}$.

	p				
dataset	1.8	1.9	2	2.25	2.5
3sources	0.84	0.82	0.79	0.76	0.74
BBCSport	0.91	0.89	0.87	0.85	0.83
Wikipedia	0.60	0.58	0.56	0.53	0.50
UCI	0.94	0.93	0.91	0.88	0.86
cora	0.71	0.69	0.66	0.62	0.60
primary school	0.89	0.85	0.82	0.77	0.75
high school	0.96	0.95	0.93	0.89	0.87

In order to investigate how the regularization parameter p influences the behavior of the methods and the accuracy of the results, we further carried out experiments with $p \neq 2$. In particular, we took into consideration both values larger and smaller than 2. We show the results for $p \in \{1.8, 1.9, 2.25, 2.5\}$, with fixed $perc = 6\%$ (resp. *Wikipedia* $perc = 18\%$). In Figs. 7 and 8, we report the average values of the objective function, and in Figs. 9 and 10, the related accuracy values. In Table 3, we present aggregated results of the objective function and the accuracy, as explain in Section 4.1. We can notice that the behavior of the methods does not change much by varying p. The GCD method performs better than the others in terms of number of flops. Looking at the values of the objective function, the CCD method seems to have a behavior very similar to the ones of the GD. Meanwhile, the RCD method performs poorly both in terms of objective function and accuracy. It is important to observe that $p \neq 2$ can lead to an improvement of the final accuracy. In Table 4, we report the maximum value of accuracy achieved in the real datasets with fixed $perc = 6\%$ (resp. $perc = 18\%$ for Wikipedia) and varing $p \in \{1.8, 1.9, 2, 2.25, 2.5\}$.

5. Conclusions

In this paper, we compared different coordinate descent methods with the standard Gradient Descent approach for the resolution of an optimization-based formulation of the Graph Semi-Supervised learning problem on multilayer hypergraphs. We performed extensive experiments on both synthetic and real world datasets, which show the faster convergence speed of suitably chosen coordinate methods with respect to the Gradient Descent approach. This fact clearly indicates that the design of tailored coordinate methods for the resolution of the considered semi-supervised learning problems represents a fruitful path to follow. In addition, we carried out an analysis replacing the standard quadratic regularization term in the objective function with a more general p − regularizer. The reported results clearly show that this modification can lead to better performances.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors would like to thank the Editors and the two Reviewers for their constructive comments and suggestions that allowed them to greatly improve the manuscript.

Appendix A. Calculations

In order to compare the gradient descent method to block coordinate descent approaches, we need to calculate the gradient of the function $\vartheta(Z)$ that we want to minimize in (5). The gradient of $\vartheta(Z)$ can be expressed as:

$$\nabla\vartheta(Z) = 2(Z-Y) + p\sum_{\ell=1}^{L}\lambda_\ell \mathscr{L}_\ell^p(Z), \tag{A.1}$$

where $\mathscr{L}_\ell^p(Z)$ is the normalized $p-$ laplacian and it is applied on each column of Z in this way:

$$\mathscr{L}_\ell^p(Z) = \left(B_\ell D_\ell^{-\frac{1}{2}}\right)^T \phi_p\left(B_\ell D_\ell^{-\frac{1}{2}} Z\right),$$

with $\phi_p(y) = |y|^{p-1}\mathrm{sgn}(y)$ component-wise.

At the beginning, we calculate $B_\ell D_\ell^{-\frac{1}{2}}\ \forall \ell \in \{1,..,L\}$ layer. Then, at each iteration k, the gradient is calculated in an iterative fashion. Break the formula of the gradient in (A.1) into two parts:

$$\nabla\vartheta(Z) = \nabla f(Z) + \nabla r_p(Z),$$

with

$$\begin{aligned}\nabla f(Z) &= 2(Z-Y),\\ \nabla r_p(Z) &= p\sum_{\ell=1}^{L}\lambda^\ell \mathscr{L}_\ell^p(Z).\end{aligned}$$

Then,

$$\begin{aligned}\nabla f\left(Z^{k+1}\right) &= \nabla f\left(Z^k\right) + 2\left(Z_{W^k}^{K+1} - Z_{W^k}^K\right),\\ \nabla r_p\left(Z^{k+1}\right) &= p\sum_{\ell=1}^{L}\lambda^\ell \mathscr{L}_\ell^p\left(Z^{k+1}\right),\end{aligned}$$

where $\mathscr{L}_\ell^p(Z^{k+1})$ can be iteratively calculated using

$$B_\ell D_\ell^{-\frac{1}{2}} Z^{K+1} = B_\ell D_\ell^{-\frac{1}{2}} Z^K + \left(B_\ell D_\ell^{-\frac{1}{2}}\right)_{W^k}\left(Z_{W^k}^{K+1} - Z_{W^k}^K\right)$$

with the appropriate subscript W^k to take just the W_j^k coordinates of column j, for all $j \in \{1,\ldots,m\}$.

A1. *Special case* $p = 2$

In this section, we discuss the special case of problem (5) with $p = 2$. The optimization problem (5) is equivalent to:

$$\min_{Z\in\mathbb{R}^{|V|\times k}} \| Z - Y \|_{(2)}^2 + \sum_{\ell=1}^{L}\lambda_\ell Z^T \overline{L}_\ell Z,$$

where $\overline{L}_\ell = I - \overline{A}_\ell$ is the normalized laplacian matrix of layer $\ell = 1,\ldots,L$ and $\overline{A}_\ell$ is the normalized adjacency matrix of layer $\ell = 1,\ldots,L$ with entries

$$(\overline{A}_\ell)_{uv} = \frac{(A_\ell)_{uv}}{\sqrt{(\delta_\ell)_u}\sqrt{(\delta_\ell)_v}}.$$

In this case, the gradient of the function to minimize can be expressed as

$$\nabla_Z\vartheta(Z) = 2(Z-Y) + \sum_{\ell=1}^{L} 2\lambda_\ell \overline{L}_\ell Z$$

and the Hessian as $2I + \sum_{\ell=1}^{L} 2\lambda_\ell \overline{L}_\ell$. In the experiments where $p = 2$, this last expression can be used in the calculation of the stepsize.

References

[1] A. Argyriou, M. Herbster, M. Pontil, Combining graph laplacians for semi–supervised learning, Adv. Neural Inf. Process. Syst. 18 (2005).
[2] F. Battiston, G. Cencetti, I. Iacopini, V. Latora, M. Lucas, A. Patania, J.-G. Young, G. Petri, Networks beyond pairwise interactions: structure and dynamics, Phys. Rep. 874 (2020) 1–92.
[3] A. Beck, The 2-coordinate descent method for solving double-sided simplex constrained minimization problems, J. Optim. Theory Appl. 162 (3) (2014) 892–919.
[4] D.P. Bertsekas, Nonlinear Programming, Athena Scientific, Belmont, MA, 1999.
[5] D. Bertsekas, J. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods, Athena Scientific, 2015.
[6] E.G. Birgin, J.M. Martínez, Block coordinate descent for smooth nonconvex constrained minimization, Comput. Optim. Appl. 83 (1) (2022) 1–27.
[7] S. Boccaletti, G. Bianconi, R. Criado, C.I. Del Genio, J. Gómez-Gardenes, M. Romance, I. Sendina-Nadal, Z. Wang, M. Zanin, The structure and dynamics of multilayer networks, Phys. Rep. 544 (1) (2014) 1–122.
[8] T. Bühler, M. Hein, Spectral clustering based on the graph p-Laplacian. Proceedings of the 26th annual international conference on machine learning, 2009, pp. 81–88.
[9] J. Calder, The game theoretic p-Laplacian and semi-supervised learning with few labels, Nonlinearity 32 (1) (2018) 301.
[10] A. Cassioli, D. Di Lorenzo, M. Sciandrone, On the convergence of inexact block coordinate descent methods for constrained optimization, Eur. J. Oper. Res. 231 (2) (2013) 274–281.
[11] O. Chapelle, B. Schölkopf, A. Zien, Semi-supervised learning. adaptive computation and machine learning, Methods 1 (1) (2010) 4–8.
[12] U. Chitra, B. Raphael, Random walks on hypergraphs with edge-dependent vertex weights. International Conference on Machine Learning, PMLR, 2019, pp. 1172–1181.
[13] P.S. Chodrow, N. Veldt, A.R. Benson, Generative hypergraph clustering: from blockmodels to modularity, Sci. Adv. 7 (28) (2021) eabh1303.
[14] A. Cristofari, F. Rinaldi, F. Tudisco, Total variation based community detection using a nonlinear optimization approach, SIAM J. Appl. Math. 80 (3) (2020) 1392–1419.
[15] A. Cristofari, An almost cyclic 2-coordinate descent method for singly linearly constrained problems, Comput. Optim. Appl. 73 (2) (2019) 411–452.
[16] A. Cristofari, A decomposition method for lasso problems with zero-sum constraint, Eur. J. Oper. Res. 306 (1) (2023) 358–369.
[17] M. De Santis, S. Lucidi, F. Rinaldi, A fast active set block coordinate descent algorithm for ℓ_1-Regularized least squares, SIAM J. Optim. 26 (1) (2016) 781–809.
[18] P. Deidda, M. Putti, F. Tudisco, Nodal domain count for the generalized graph p-Laplacian, Appl. Comput. Harmon. Anal. 64 (2023) 1–32.
[19] A. Demiriz, K.P. Bennett, Optimization approaches to semi-supervised learning. Complementarity: Applications, Algorithms and Extensions, Springer, 2001, pp. 121–141.
[20] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2002) 201–213.
[21] H.-C. Dong, Y.-F. Li, Z.-H. Zhou, Learning from semi-supervised weak-label data. Proceedings of the AAAI Conference on Artificial Intelligence volume 32, 2018.
[22] A. El Alaoui, X. Cheng, A. Ramdas, M.J. Wainwright, M.I. Jordan, Asymptotic behavior of\ell_p-based laplacian regularization in semi-supervised learning. Conference on Learning Theory, PMLR, 2016, pp. 879–906.
[23] D. Eswaran, S. Günnemann, C. Faloutsos, D. Makhija, M. Kumar, Zoobp: belief propagation for heterogeneous networks, Proc. VLDB Endowment 10 (5) (2017) 625–636.
[24] M. Flores, J. Calder, G. Lerman, Analysis and algorithms for ℓ_p-based semi-supervised learning on graphs, Appl. Comput. Harmon. Anal. 60 (2022) 77–122.
[25] A. Ghaffari-Hadigheh, L. Sinjorgo, R. Sotirov, On convergence of a q-random coordinate constrained algorithm for non-convex problems, arXiv preprint arXiv: 2210.09665 (2022).
[26] L. Grippo, M. Sciandrone, Globally convergent block-coordinate techniques for unconstrained optimization, Optim. Methods Softw. 10 (4) (1999) 587–637.
[27] L. Grippo, M. Sciandrone, On the convergence of the block nonlinear gauss–Seidel method under convex constraints, Oper. Res. Lett. 26 (3) (2000) 127–136.
[28] E. Gujral, E.E. Papalexakis, Smacd: semi-supervised multi-aspect community detection. Proceedings of the 2018 SIAM International Conference on Data Mining, SIAM, 2018, pp. 702–710.
[29] M. Gürbüzbalaban, A. Ozdaglar, N.D. Vanli, S.J. Wright, Randomness and permutations in coordinate descent methods, Math. Program. 181 (2) (2020) 349–376.
[30] M. Hein, S. Setzer, L. Jost, S.S. Rangapuram, The total variation on hypergraphs-learning on hypergraphs revisited, Adv. Neural Inf. Process. Syst. 26 (2013).
[31] P.W. Holland, K.B. Laskey, S. Leinhardt, Stochastic blockmodels: first steps, Soc. Netw. 5 (2) (1983) 109–137.
[32] R. Ibrahim, D.F. Gleich, Local hypergraph clustering using capacity releasing diffusion, PLoS ONE 15 (12) (2020) e0243485.
[33] M. Karasuyama, H. Mamitsuka, Multiple graph label propagation by sparse integration, IEEE Trans. Neural Netw. Learn. Syst. 24 (12) (2013) 1999–2012.
[34] H. Karimi, J. Nutini, M. Schmidt, Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. Joint European conference on machine learning and knowledge discovery in databases, Springer, 2016, pp. 795–811.
[35] R. Kyng, A. Rao, S. Sachdeva, D.A. Spielman, Algorithms for Lipschitz learning on graphs. Conference on Learning Theory, PMLR, 2015, pp. 1190–1223.
[36] C.-J. Lin, On the convergence of the decomposition method for support vector machines, IEEE Trans. Neural Netw. 12 (6) (2001) 1288–1298.
[37] S. Lucidi, L. Palagi, A. Risi, M. Sciandrone, A convergent decomposition algorithm for support vector machines, Comput. Optim. Appl. 38 (2) (2007) 217–234.
[38] Z.-Q. Luo, P. Tseng, On the convergence of the coordinate descent method for convex differentiable minimization, J. Optim. Theory Appl. 72 (1) (1992) 7–35.
[39] M. Magnani, O. Hanteer, R. Interdonato, L. Rossi, A. Tagarelli, Community detection in multiplex networks, ACM Comput. Surv. (CSUR) 54 (3) (2021) 1–35.
[40] P. Mercado, F. Tudisco, M. Hein, Generalized matrix means for semi-supervised learning with multilayer graphs, arXiv:1910.13951 (2019).
[41] B. Nadler, N. Srebro, X. Zhou, Semi-supervised learning with the graph laplacian: the limit of infinite unlabelled data, Adv. Neural Inf. Process. Syst. 22 (2009) 1330–1338.
[42] I. Necoara, A. Patrascu, A random coordinate descent algorithm for optimization problems with composite objective function and linear coupled constraints, Comput. Optim. Appl. 57 (2) (2014) 307–337.
[43] I. Necoara, Y. Nesterov, F. Glineur, Random block coordinate descent methods for linearly constrained optimization over networks, J. Optim. Theory Appl. 173 (1) (2017) 227–254.
[44] Y. Nesterov, Efficiency of coordinate descent methods on huge-scale optimization problems, SIAM J. Optim. 22 (2) (2012) 341–362.
[45] F. Nie, J. Li, X. Li, et al., Parameter-free auto-weighted multiple graph learning: a framework for multiview clustering and semi-supervised classification. IJCAI, 2016, pp. 1881–1887.
[46] D. Zhou, O. Bousquet, T. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency, in: S. Thrun, L. Saul, B. Schölkopf (Eds.), Advances in Neural Information Processing Systems volume 16, MIT Press, 2003.https://proceedings.neurips.cc/paper/2003/file/87682805257e619d49b8e0dfdc14affa-Paper.pdf
[47] J. Nutini, M. Schmidt, I. Laradji, M. Friedlander, H. Koepke, Coordinate descent converges faster with the Gauss-Southwell rule than random selection. International Conference on Machine Learning, PMLR, 2015, pp. 1632–1641.
[48] J. Nutini, I. Laradji, M. Schmidt, Let'S make block coordinate descent converge faster: faster greedy rules, message-Passing, active-Set complexity, and superlinear convergence, J. Mach. Learn. Res. 23 (131) (2022) 1–74.
[49] A. Patrascu, I. Necoara, Efficient random coordinate descent algorithms for large-scale structured nonconvex optimization, J. Global Optim. 61 (1) (2015) 19–46.
[50] K. Prokopchik, A.R. Benson, F. Tudisco, Nonlinear Feature Diffusion on Hypergraphs. International Conference on Machine Learning, PMLR, 2022, pp. 17945–17958.
[51] Z. Qu, P. Richtárik, Coordinate descent with arbitrary sampling II: expected separable overapproximation, Optim. Method. Softw. 31 (5) (2016) 858–884.
[52] M. Razaviyayn, M. Hong, Z.-Q. Luo, A unified convergence analysis of block successive minimization methods for nonsmooth optimization, SIAM J. Optim. 23 (2) (2013) 1126–1153.

[53] S. Reddi, A. Hefny, C. Downey, A. Dubey, S. Sra, Large-scale randomized-coordinate descent methods with non-separable linear constraints, Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI) (2015).
[54] P. Richtárik, M. Takáč, Distributed coordinate descent method for learning with big data, J. Mach. Learn. Res. 17 (1) (2016) 2657–2681.
[55] P. Richtárik, M. Takáč, Parallel coordinate descent methods for big data optimization, Math. Program. 156 (2016) 433–484.
[56] P. Richtárik, M. Takáč, Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function, Math. Program. 144 (1) (2014) 1–38.
[57] S. Salzo, S. Villa, Parallel random block-coordinate forward–backward algorithm: a unified convergence analysis, Math. Program. 193 (1) (2022) 225–269.
[58] R. Sargent, D.J. Sebastian, On the convergence of sequential minimization algorithms, J. Optim. Theory Appl. 12 (6) (1973) 567–575.
[59] D. Slepcev, M. Thorpe, Analysis of p-Laplacian regularization in semisupervised learning, SIAM J. Math. Anal. 51 (3) (2019) 2085–2120.
[60] Z. Song, X. Yang, Z. Xu, I. King, Graph-based semi-supervised learning: acomprehensive review, IEEE Trans. Neural Netw. Learn. Syst. (2022).
[61] A. Subramanya, P.P. Talukdar, Graph-based semi-supervised learning, Synthesis Lect. Artif. Intell. Mach. Learn. 8 (4) (2014) 1–125.
[62] P. Tseng, S. Yun, A coordinate gradient descent method for nonsmooth separable minimization, Math. Program. 117 (1) (2009) 387–423.
[63] P. Tseng, S. Yun, Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization, J. Optim. Theory Appl. 140 (3) (2009) 513–535.
[64] K. Tsuda, H. Shin, B. Schölkopf, Fast protein classification with multiple networks, Bioinformatics 21 (suppl_2) (2005) ii59–ii65.
[65] F. Tudisco, M. Hein, A nodal domain theorem and a higher-order cheeger inequality for the graph *p*-laplacian, EMS J. Spectral Theory 8 (2018) 883–908.
[66] F. Tudisco, P. Mercado, M. Hein, Community detection in networks via nonlinear modularity eigenvectors, SIAM J. Appl. Math. 78 (2018) 2393–2419.
[67] F. Tudisco, A.R. Benson, K. Prokopchik, Nonlinear higher-order label spreading. Proceedings of The Web Conference, 2021, p. toappear.
[68] F. Tudisco, D. Zhang, Nonlinear spectral duality, arxiv:2209.06241 (2022).
[69] N. Veldt, A.R. Benson, J. Kleinberg, Minimizing localized ratio cut objectives in hypergraphs. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1708–1718.
[70] S. Venturini, A. Cristofari, F. Rinaldi, F. Tudisco, A variance-aware multiobjective louvain-like method for community detection in multiplex networks, J. Complex Netw. 10 (6) (2022) cnac048.
[71] S. Venturini, A. Cristofari, F. Rinaldi, F. Tudisco, Learning the right layers a data-driven layer-aggregation strategy for semi-supervised learning on multilayer graphs. Proceedings of the 40th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, volume 202, PMLR, 2023, pp. 35006–35023.
[72] J.J. Whang, R. Du, S. Jung, G. Lee, B. Drake, Q. Liu, S. Kang, H. Park, MEGA: multi-view semi-supervised clustering of hypergraphs, Proc. VLDB Endowment 13 (5) (2020) 698–711.
[73] S.J. Wright, Coordinate descent algorithms, Math. Program. 151 (1) (2015) 3–34.
[74] H. Yin, A.R. Benson, J. Leskovec, D.F. Gleich, Local higher-order graph clustering. Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, 2017, pp. 555–564.
[75] C. Zhang, S. Hu, Z.G. Tang, T.H.H. Chan, Re-revisiting learning on hypergraphs: confidence interval and subgradient method. International Conference on Machine Learning, PMLR, 2017, pp. 4026–4034.
[76] D. Zhou, J. Huang, B. Schölkopf, Learning with hypergraphs: clustering, classification, and embedding, Adv. Neural Inf. Process. Syst. 19 (2006).
[77] D. Zhou, C.J.C. Burges, Spectral clustering and transductive learning with multiple views. Proceedings of the 24th international conference on Machine learning, 2007, pp. 1159–1166.
[78] X. Zhou, M. Belkin, Semi-supervised learning by higher order regularization. Proceedings of the fourteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2011, pp. 892–900.
[79] X. Zhu, Z. Ghahramani, J.D. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions. Proceedings of the 20th International conference on Machine learning (ICML-03), 2003, pp. 912–919.