A reductive approach to hypergraph clustering: An application to image segmentation

Aurélien Ducournau, Alain Bretto, Soufiane Rital, Bernard Laget

A reductive approach to hypergraph clustering: An application to image segmentation

Aurélien Ducournau [a], Alain Bretto [c,b,*], Soufiane Rital [b], Bernard Laget [a]

[a] ENISE-DIPI, 58 rue Jean Parot, 42023 Saint-Etienne Cedex 2, France
[b] Telecom ParisTech (ENST), TSI, CNRS-LTCI, 46 rue Barrault, F-75634 Paris Cedex 13, France
[c] GREYC-CNRS UMR 6072, Campus Côte de Nacre, Boulevard du Maréchal Juin, BP 5186, 14032 Caen Cedex, France

A B S T R A C T

In the last few years, hypergraph-based methods have gained considerable attention in the resolution of real-world clustering problems, since such a mode of representation can handle higher-order relationships between elements compared to the standard graph theory. The most popular and promising approach to hypergraph clustering arises from concepts in spectral hypergraph theory [53], and clustering is configured as a hypergraph cut problem where an appropriate objective function has to be optimized. The spectral relaxation of this optimization problem allows to get a clustering that is close to the optimum, but this approach generally suffers from its high computational demands, especially in real-world problems where the size of the data involved in their resolution becomes too large. A natural way to overcome this limitation is to operate a reduction of the hypergraph, where spectral clustering should be applied over a hypergraph of smaller size. In this paper, we introduce two novel hypergraph reduction algorithms that are able to maintain the hypergraph structure as accurate as possible. These algorithms allowed us to design a new approach devoted to hypergraph clustering, based on the multilevel paradigm that operates in three steps: (i) hypergraph reduction; (ii) initial spectral clustering of the reduced hypergraph and (iii) clustering refinement. The accuracy of our hypergraph clustering framework has been demonstrated by extensive experiments with comparison to other hypergraph clustering algorithms, and have been successfully applied to image segmentation, for which an appropriate hypergraph-based model have been designed. The low running times displayed by our algorithm also demonstrates that the latter, unlike the standard spectral clustering approach, can handle datasets of considerable size.

1. Introduction

Graph/hypergraph based methods have played an important role in Computer Vision and Pattern Recognition (CVPR) due to their ability to represent relational patterns. For an overview of recent literature, we refer the reader to a number of special issues that appeared on the subject [45,48] and some interesting and recent articles [46,4,32,10].

In many situations, a graph-based representation is incomplete, as only binary relations between nodes can be represented through graph edges. An extension is provided by hypergraphs, where each edge is a subset of the set of nodes [5,9]. Hence higher-order relations between nodes can be directly modeled in a hypergraph, by the means of hyperedges. For instance, Fig. 1 presents a graph (a) and a hypergraph (b) that model the same image. The graph consists of a Region Adjacency Graph (RAG) in which each vertex represents a distinct region of the image, and two vertices are linked by a graph edge if their corresponding regions are neighbors in the image. One can see that this representation can only involve pairwise neighboring relationships between regions, as the human eye clearly identifies that the region 3, for example, has common boundaries with regions 2 and 4. This kind of information is not provided by graph edges, while they can easily be represented through hyperedges. If you look at Fig. 1(b), the neighboring relationship between regions 2, 3 and 4 is modeled by the means of the red hyperedge. The hypergraph displayed in Fig. 1(b) only presents regions adjacencies of degree three to clarify the hypergraph visualization, but hypergraphs can deal with hyperedge of arbitrary sizes. More examples showing the benefits of hypergraphs in CVPR domain can be found in [9].

A large body of theoretical work on hypergraphs has been published in various domains such as pattern recognition, VLSI design, database design, software engineering, parallel scientific

* Corresponding author at: GREYC-CNRS UMR 6072, Campus Côte de Nacre, Boulevard du Maréchal Juin, BP 5186, 14032 Caen Cedex, France.
E-mail addresses: aurelien.ducournau@enise.fr (A. Ducournau), alain.bretto@unicaen.fr (A. Bretto), soufiane.rital@telecom-paristech.fr (S. Rital), bernard.laget@enise.fr (B. Laget).

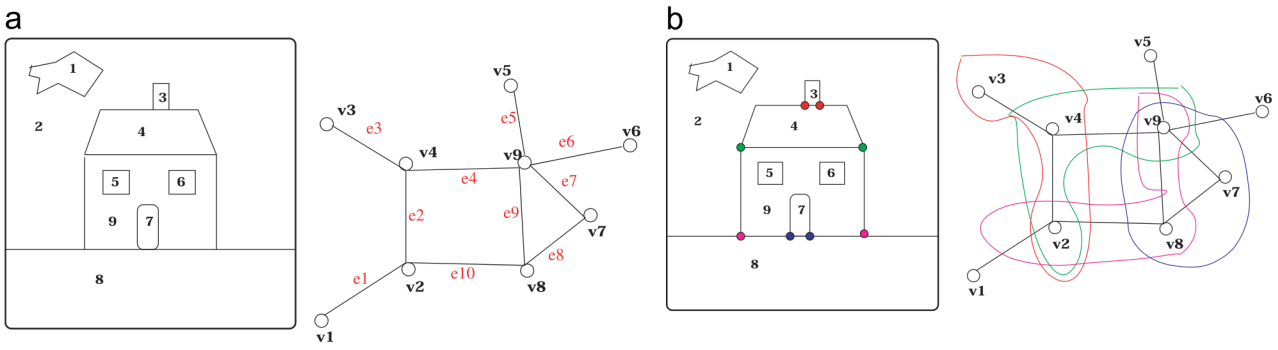

Fig. 1. Graph (a) and hypergraph (b) examples. The graph in (a) illustrates a Regions Adjacency Graph (RAG). The hypergraph (b) presents a RAG and also the vertices that share some corners. (a) Graph: $G=(V,e)$; (b) hypergraph $H=(V,E)$.

computing and machine learning. Hypergraphs have also been recently introduced in computer vision, and the next section presents an overview of the contributions of hypergraph theory in this field.

1.1. Hypergraph theory and computer vision: a short overview

To the best of our knowledge, the first attempt for representing visual objects using hypergraphs dates back to Wong et al. In [50], the authors defined a framework for 3D object recognition. They used a 3D object model based on a hypergraph representation. Object synthesis and recognition tasks are performed by merging and partitioning the hyperedge and vertex set, and the hypergraphs are not characterized in a mathematically consistent way.

In [5], Bretto et al. introduced a hypergraph model for image representation by defining a new Image Neighborhood Hypergraph (INH). The INH model have then been exploited to successfully solve the problem of image segmentation by finding intersecting families to detect grayscale homogeneous regions. Using the INH representation, Rital et al. expanded the hypergraph applications to other computer vision problems. Using hypergraph properties, they proposed noise removal [43] and edge detection [44] solutions. In this paper, the INH model serves as a basis for image data representation. Bunke et al. [9] have developed a hypergraph matching algorithm for object recognition, where consistency checks are conducted on hyperedges. The computational paradigm underlying their method is based on tree search operations. Since Chung's [15] definition of the Laplacian matrix for k-uniform hypergraphs, there have been several attempts to develop matrix representations of hypergraphs [35,52]. These hypergraph representations have found widespread applications in categorical data analysis. In [1], Agarwal et al. related hypergraph based learning to graph based learning. They approximated the hypergraph with a clique averaging and used a graph cut algorithm to partition an image database. The graph algorithm is based on the notion of the normalized graph Laplacian matrix. In [53], Zhou et al. generalized this notion to the hypergraph case, in order to provide a spectral hypergraph clustering algorithm that showed encouraging classification results on categorical data. A supervised bi-partitioning approach using Zhou algorithm is used by Ding et al. [20] for object-background segmentation problems. Ren et al. have showed that a matrix representation is also suitable for image processing [40], and have proposed an improved hypergraph Laplacian matrix based on the developments of Zhou et al.'s method [53]. However, this method is based on a relatively impoverished spectral characterization and overlooks much of the details of the hypergraph structure. Recently, other applications have emerged and expanded the application area of hypergraphs. Gillibert et al. used another model based on rectangle hyperedge [8] to define an efficient lossless image compression algorithm. In [29], Huang et al. used a hypergraph in a video segmentation application. The hypergraph theory is often used as a solution for partitioning large masses of data, especially in VLSI design [31], parallel scientificcomputing [11] and database design [33].

1.2. Graph and hypergraph clustering

Clustering is closely related to unsupervised learning in pattern recognition systems [22]. A basic task in unsupervised learning is to group patterns on the basis of a similarity (or dissimilarity) criteria where groups (or clusters) are sets of similar patterns. In graph clustering, graph theory can provide the necessary definitions and mathematical formalism, resulting in an important support for the analysis of clustering models. According to [45], graph clustering groups vertices of a graph into clusters, based on the edge structure of the graph. The resulting vertex partition should have the property that within each cluster the vertices are highly connected whereas there are only few edges between clusters. Reviews on graph clustering can be found in [45,26,38]. The main graph clustering formulations are based on graph cut and partitioning problems [13,46]. An alternative to heuristically solve these problems is to use spectral clustering algorithms. The basic idea is to construct a weighted graph from the initial data set where each node represents a pattern and each weighted edge simply takes into account the similarity between two patterns. In this framework the clustering problem can be seen as a graph cut problem, which can be tackled by means of the spectral graph theory. The core of this theory is the eigenvalue decomposition of the Laplacian matrix of the weighted graph obtained from data. In fact, there is a close relationship between the second smallest eigenvalue of the Laplacian and the graph cut [25].

Like graphs, hypergraphs may be partitioned such that a cut metric is minimized. The most extensive and large scale use of hypergraph partitioning algorithms, however, occurs in the field of VLSI design and synthesis. A typical application involves the partitioning of large circuits into k roughly equally sized parts in a manner that minimizes the connectivity between the parts. The circuit elements are the vertices of the hypergraph and the nets that connect these circuit elements are the hyperedges [2]. Several serial and parallel hypergraph partitioning techniques have been extensively studied [31,49,14] and tools support exists (e.g. hMETIS [31], PaToH [12], Parkway [49] and PT-Scotch [14]). These partitioning techniques showed a very great efficiency in

distributed databases and VLSI circuit fields. For an overview of recent heuristic hypergraph partitioning algorithms, we refer the reader to [3]. Hypergraph cut metrics provide a more accurate model than graph partitioning in many cases of practical interest. For example, in the row-wise decomposition of a sparse matrix for parallel matrix–vector multiplication, a hypergraph model provides an exact measure of communication cost, whereas a graph model can only provide an upper bound [12,49]. Recently, Zhou et al. generalized the normalized graph Laplacian matrix to the hypergraph case that have showed encouraging results on categorical data [53]. The spectral methods have been shown very effective for solving the partitioning problem, but it is computationally expensive, especially with massive grid graphs or hypergraphs. On the other hand, heuristic methods offer linear-time execution algorithms but provide under-optimal solutions. Consequently, the partitioning algorithms must be carefully designed and implemented to increase the quality of the optimization.

1.3. Our contribution

Most contributions using hypergraph are focused on hypergraph representation and/or the use of hypergraph properties. The drawbacks of most of these approaches are twofold: the loss of information and the computational complexity resulting respectively on how a hypergraph representation is computed and when the hypergraph properties are exploited. Loss of information comes from that most of the approaches approximate a hypergraph by a graph such as clique-graph, line graph and then use graph algorithms to solve their problems [1,29]. The computational complexity is generally due to the amount of nodes embedded in the hypergraph, and for most of them exploiting directly the properties of the initial hypergraph representation without including a reduction strategy is difficult to apply in practice. Moreover, hypergraphs offer a representation that is richer in knowledge than graphs, and in particular the number of relationships is not bounded. Recently, in computer vision problems, many authors operate in the superpixels [28,29] domain rather than the pixels [53,20] one in order to minimize the size of the data. Often this reduction step is ignored and entrusted to other algorithms [46,16,23].

In this paper, we consider these two problems, and we propose a new strategy for hypergraph clustering. We have clearly identified a new approach to perform clustering using the multilevel paradigm developed initially by Karypis [31]. The proposed approach operates in three phases: hypergraph reduction, initial clustering, and clustering refinement. During the reduction phase, a sequence of successive smaller hypergraphs is constructed. The members of the sequence approximate the original hypergraph at successive coarser scales of resolution. This is the most important phase of the multilevel paradigm, and its overall success relies on being able to find reasonable methods for obtaining these reduced hypergraphs. Consequently, the core of our hypergraph clustering framework is based on the definition of a brand new hypergraph reduction process that is the main contribution of our paper. The algorithm has been successfully applied to solve a widely discussed problem in computer vision: image segmentation, for which a suitable hypergraph model is used.

A preliminary version of this work can be found in [21,7]. In [21], we developed a hypergraph clustering approach that takes advantage from both multilevel and spectral partitioning techniques. This algorithm exploited an INH model to provide an image segmentation application. In [7], a new hypergraph reduction algorithm has been designed and exploited for image superpixels generation. The superpixels served as a basis for a satellite image content classification process that has the advantage to handle high-resolution images. In this paper, we extend the results introduced in [7] by proposing a new hypergraph reduction algorithm, and studying its mathematical properties. These properties allowed us to develop our hypergraph clustering algorithm by following the idea introduced in [21], as well as its application to image segmentation.

The remainder of this paper is organized as follows: we first introduce some basic notions on hypergraphs in Section 2. The proposed hypergraph reduction algorithms are introduced in Section 4, as well as some useful mathematical properties. In Section 5, we define the major steps of our hypergraph clustering algorithm. In Section 6, an application of our clustering framework to color image segmentation is given. Experimental results are illustrated in Section 7. Conclusions and perspectives are discussed in Section 8.

2. Preliminaries

Let $I=\{1,2,\ldots,m\}$ with $m>0$. A hypergraph $H=(V;E)$ on a set V is a family $E=(e_i)_{i\in I}$ of nonempty subsets of V called hyperedges, with $\bigcup_{i\in I}e_i=V$. A weighted hypergraph is a hypergraph that has a positive number $w(e)$ associated with each hyperedge e, called the weight of hyperedge e. Denote a weighted hypergraph by $H=(V;E;w)$. For a vertex $v\in V$, its degree is defined by $\sum_{e\in E|v\in e}w(e)$. The cardinality of V is denoted by $|V|$. For a hyperedge $e\in E$, its degree is defined to be $\delta(e)=|e|$. A hyperedge e_i is isolated if and only if $\forall j\in I, j\neq i$, if $e_i\cap e_j\neq\emptyset$, then $e_j\subseteq e_i$.

A hypergraph H can be represented by a $|V|\times|E|$ matrix $\mathcal{H}$ with entries $h(i,j)=1$ if $v_i\in e_j$ and 0 otherwise, called the incidence matrix of H. Then $d(v_i)=\sum_{e_j\in E}w(e_j)h(i,j)$ and $\delta(e_j)=\sum_{v_i\in V}h(i,j)$. Let D_v and D_e denote the diagonal matrices containing the vertex and hyperedge degrees respectively, and let W denote the diagonal matrix containing the weights of hyperedges. Then the adjacency matrix A of hypergraph H is defined as $A=\mathcal{H}W\mathcal{H}^T-D_v$, where $\mathcal{H}^T$ is the transpose of $\mathcal{H}$.

Given a subset of vertices $S\subset V$, the *volume* of S is defined by $\mathrm{vol}S=\sum_{v\in S}d(v)$. The complementary of S is noted S^c. We say that a hyperedge e is *cut* if it contains vertices from S and S^c simultaneously. So, we define the *hyperedge boundary* ∂S as the set of hyperedges containing vertices from S which are cut, i.e. $\partial S=\{e\in E|e\cap S\neq\emptyset, e\cap S^c\neq\emptyset\}$. Thus, the volume of ∂S is given by $\mathrm{vol}\partial S=\sum_{e\in\partial S}w(e)|e\cap S||e\cap S^c|/\delta(e)$.

The line graph $L(H)=(E;A)$ of a hypergraph H is the graph whose vertex set is the set of hyperedges of the hypergraph, with two edges adjacent when they have a nonempty intersection. In other words, the line graph of a hypergraph is the intersection graph of a family of finite sets. It is a generalization of the line graph of a graph.

Given a graph $G=(V;E)$, where V is a set of vertices, and E is a set of unordered pairs of members of V called edges. The hypergraph having the vertices of G as vertices and the neighborhood of these vertices as hyperedges (including these vertices) is called the neighborhood hypergraph of graph G. To each G we can associate a neighborhood hypergraph $H_G=(V;(E_v=\{v\}\cup\Gamma(v)))$ where $\Gamma(v)=\{v'\in V,\{v,v'\}\in E\}$.

Let $H_1=(V_1;E_1)$ and $H_2=(V_2;E_2)$ be two hypergraphs. A map f from V_1 to V_2 is a *morphism* or *homomorphism* if it verifies the following properties:

$$e\in E_1\Longrightarrow f(e)=\{f(v),v\in e\}\subset a\in E_2 \tag{1}$$

Let $H=(V;E)$ be a hypergraph, a *partial hypergraph* of H is a subfamily $(e_j)_{j\in J}$ of $(e_i)_{i\in I}$, with $J\subseteq I$. A *subhypergraph* on X of the

hypergraph H is the hypergraph $H(X)=(X;(e_i\cap X\neq\emptyset)_{i\in I})$, (with $X\subseteq V$).

We say that there is a hyperpath between vertices v_1 and v_k when there is an alternative sequence of distinct vertices and hyperedges $v_1,e_1,v_2,e_2,\ldots,e_{k-1},v_k$ such that $\{v_i,v_{i+1}\}\subseteq e_i$ for $1\leq i\leq k-1$. A hypergraph is connected if there is a hyperpath for every pair of vertices.

The excentricity $\epsilon(v)$ of a vertex v in a connected graph G is the maximum graph distance between v and any other vertex u of G.

3. Previous work on hypergraph reduction

As stated in Section 1.3, the reduction process is the most important part of a multilevel partitioning algorithm, and the overall success of the algorithm depends on the ability of the reduction step to preserve as much as possible the structural properties of the initial hypergraph. In the following, some previous work in this direction is presented. For more information, we refer the reader to [31].

Edge Coarsening (*EC*). The simplest way to group the vertices is to select pairs of vertices that are present in the same hyperedges, as illustrated in Fig. 2(a). These pairs of vertices can be formed by finding a maximal matching of the vertices that are connected via hyperedges, in which each hyperedge has been replaced by its clique representation. The vertices are visited in a random order. For each vertex v, all unmatched vertices that belong to hyperedges incident to v are considered, and the one that is connected via the edge with the largest weight is matched with v.

Hyperedge Coarsening (*HC*). In this scheme, a set of independent hyperedges is selected and the vertices that belong to individual hyperedges are contracted together, as illustrated in Fig. 2(b). The list of hyperedges is sorted by descending weights, giving favor to the selection of hyperedges with large weights.

Modified Hyperedge Coarsening (*MHC*). After the hyperedges to be contracted have been selected using the HC scheme, the list of hyperedges is traversed again, and for each hyperedge that has not yet been contracted, the vertices that do not belong to any other contracted hyperedge are contracted together (see Fig. 2(c)).

First Choice (*FC*). The FC coarsening scheme is derived from the EC coarsening scheme by relaxing the requirement that a vertex is matched only with another unmatched vertex. Specifically, in the FC coarsening scheme, the vertices are again visited in a random order. However, for each vertex v, all vertices (both matched and unmatched) that belong to hyperedges incident to v are considered, and the one that is connected via the edge with the largest weight is matched with v, breaking ties in favor of unmatched vertices. As a result, each group of vertices to be merged together can contain an arbitrarily large number of vertices.

There are other coarsening methods used to reduce the hypergraph size. For example, the Greedy First-Choice (GFC) scheme, and the Hybrid First-Choice (HFC) scheme. In the GFC scheme vertices are grouped based on the FC scheme, but the grouping is biased in favor of faster reduction in the number of the hyperedges that remain in the coarse hypergraphs, while HFC is a combination of the FC and GFC schemes. For more information about these coarsening approaches, see [31].

4. The proposed Hypergraph Reduction (HR) algorithms

In this section, our proposed hypergraph reduction algorithms are described and some basic properties are given.

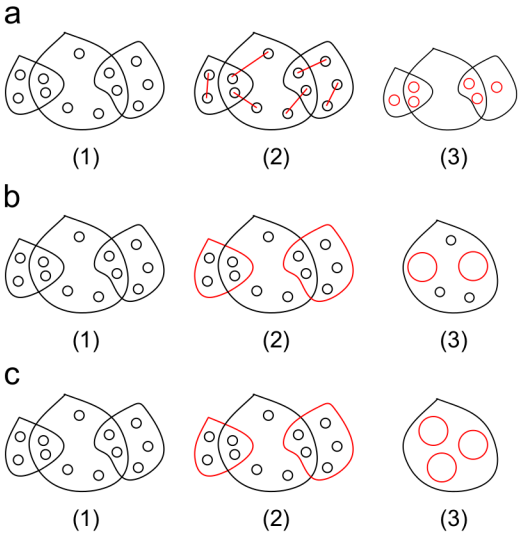

Fig. 2. Hypergraph reduction approaches developed by Karypis et al. [31]. (a) Edge Coarsening (EC): independent pairs of vertices are selected (2) and both vertices of a single pair are merged together (3). (b) Hyperedge Coarsening (HC): a set of independent hyperedges is selected (2) and all vertices that belong to the same hyperedge are merged together (3). (c) Modified Hyperedge Coarsening (MHC): after applying a HC coarsening (2 and 3), all vertices that has not been merged yet and that belong to the same hyperedge are merged together. Here the three vertices that belong to the central hyperedge are merged after the HC reduction step (3).

This section also provides the mathematical formalism and proofs that allow the utilization of a reduction algorithm in a partitioning manner such as the proposed hypergraph clustering framework.

4.1. The proposed HR using Intersecting Hyperedges (HR-IH) algorithm

4.1.1. Basic concept

In this section, we illustrate a hypergraph reduction algorithm which reduces a simple hypergraph. The full proposed hypergraph reduction algorithm of $H=(V;E)$ is described in Algorithm 1 for a given order on E. The basic idea of the proposed algorithm can be summarized as follows:

Step 1. For a given order on E, we compute the set of intersecting hyperedges W of H. For each hyperedge $e_i\in E$, we generate W_{e_i} as the set of hyperedges intersecting with e_i, i.e. $W_{e_i}=\{e_j\in E|e_i\cap e_j\neq\emptyset\}$. $W=\bigcup_{\forall e_i\in E}\{W_{e_i}\}$ is the set of intersecting hyperedges.

Step 2. From W, we keep only a subset B of W that covers the hypergraph H, i.e. that every hyperedge in E must have a nonempty intersection with at least one W_{e_i} from B. More formally we compute $B\subseteq W$ such as $\bigcup_{\forall W_{e_i}\in B}W_{e_i}=E$.

Step 3. From B, we generate the Reduced Hypergraph $RH=(RV;RE)$. Each W_{e_i} of B stand for a vertex w_{e_i} of RH. From RH and using the W_{e_i}, the set of hyperedges RE is generated in the following way: each vertex $w_{e_i}\in RV$ (that corresponds to a set of intersecting hyperedges W_{e_i}) will create an hyperedge that will contain all the vertices w_{e_j} for which the corresponding W_{e_j} shares at least one hyperedge with W_{e_i}.

Algorithm 1. HR-IH: Hypergraph Reduction using Intersecting Hyperedges.

Data: $H=(V;E=\{e_1,e_2,\ldots,e_m\})$, E is ordered.
Result: $RH=(RV;RE)$ be the reduced hypergraph of H.

begin
$W \coloneqq \emptyset$;
Step 1. The set of intersecting hyperedges;
foreach $e_i \in E$ **do**
$W_{e_i} \coloneqq \emptyset$;
foreach $e_j \in E$ **do**
if $e_i \cap e_j \neq \emptyset$ **then**
$W_{e_i} \coloneqq W_{e_i} \cup \{e_j\}$;
end
end
$W \coloneqq W \cup \{W_{e_i}\}$;
end
$B \coloneqq \emptyset$; $i \coloneqq 1$;
Step 2. The covering of the set of intersecting hyperedges;
while $E \neq \emptyset$ **do**
$U \coloneqq E \backslash W_{e_i}$;
if $|U| < |E|$ **then**
$B \coloneqq B \cup \{W_{e_i}\}$;
end
$E \coloneqq E \backslash W_{e_i}$;
$i \coloneqq i+1$;
end
Step 3. The *RH* generation;
$RV \coloneqq \emptyset$;
foreach $W_{e_i} \in B$ **do**
$RV \coloneqq RV \cup \{w_{e_i}\}$;
end
The set of hyperedges of *RE*;
$RE \coloneqq \emptyset$;
foreach $W_{e_i} \in B$ **do**
$A_{e_i} \coloneqq \emptyset$;
foreach $W_{e_j} \in B$ **do**
if $W_{e_i} \cap W_{e_j} \neq \emptyset$ **then**
$A_{e_i} \coloneqq A_{e_i} \cup \{w_{e_j}\}$
end
end
$RE \coloneqq RE \cup \{A_{e_i}\}$;
end
$RH \coloneqq (RV; RE)$
end

4.1.2. The recursive reduction process

The hypergraph reduction algorithm can be applied recursively. We define by induction the following process:

(i) $R^0H = H$
(ii) $R^{i+1}H = RR^iH$, $i \geq 1$.

This procedure creates successive smaller hypergraphs, each hypergraph R^iH should be denoted by its *level i*. By construction, the initial hypergraph H is at level 0 ($R^0H = H$). In the following, if no indication is given about the level corresponding to a hypergraph R^iH, we will note $H = R^iH$ and $R^{i+1}H = RH$ for the sake of simplicity. An example of the algorithm HR-IH is presented in Fig. 3. We reduce the initial hypergraph using three different hyperedge orders: $\{e_1,e_2,e_3\}$, $\{e_2,e_1,e_3\}$ and $\{e_3,e_2,e_1\}$. Each column in Fig. 3 represents one distinct order. We can see after a second reduction in the first and third columns that the reduction converges to a unique solution (modulo the linear order). The HR-IH algorithm presents several interesting properties that will be studied in the next section.

4.1.3. Basic properties

Proposition 1. *The algorithm creates a neighborhood hypergraph; its complexity is in $O(m^2)$, where m is the cardinality of the hyperedge set of the hypergraph.*

Proof. We can build a graph $\Gamma = (RV; A)$ in the following way:

1. The set of vertices is *RV*.
2. Let $w_{e_i}, w_{e_j} \in RV$, we put an edge between w_{e_i} and w_{e_j} iff $W_{e_i} \cap W_{e_j} \neq \emptyset$, (except when $i=j$).

Let A_{e_i} be a hyperedge of *RH*, $A_{e_i} = \{w_{e_i}; w_{e_j}$, such that $W_{e_i} \cap W_{e_j} \neq \emptyset\}$. Consequently $A_{e_i} = \{w_{e_i}\} \cup \Gamma(w_{e_i})$.

Now let $w_{e_i} \in RV$; $w_{e_j} \in \Gamma(w_{e_i}) \Longleftrightarrow W_{e_i} \cap W_{e_j} \neq \emptyset \Longleftrightarrow \{w_{e_i}\} \cup \Gamma(w_{e_i}) = A_{e_i}$.

Standard algorithms can compute the intersection of two sets S_1 and S_2 with a complexity $O(|S_1| + |S_2|)$. If we assume that the cardinality of any hyperedge is bounded by a constant *M*, then the intersection of two hyperedges e_i and e_j can be built in $O(M)$, then $O(1)$. The complexity of the algorithm is also bounded by the Step 1, which computes the set of intersecting hyperedges and consists of a double loop over the whole set of hyperedges. Consequently, the complexity of the algorithm is in $O(m^2)$. □

Since *E* is linearly ordered *B* is also linearly ordered. This order will be called Reduction Algorithm Order (*RAO*). This one is linear: $e_i \leq e_j \Longleftrightarrow W_{e_i} \leq_{RAO} W_{e_j}$. So $(B; \leq_{RAO})$ is a poset totally ordered. We will denote by $V(W_{e_i}) = \bigcup_{e_j \in W_{e_i}} \{v; v \in e_j\}$.

Proposition 2. *Let $H = (V; E)$ and $RH = (RV; RE)$ be its reduction, then there is a morphism from H to RH.*

Proof. Let *h* be defined by

$h : V \longrightarrow B$

$$v_i \mapsto \min_{j \in \{1,2,\ldots|B|\}} \{W_{e_j}, v_i \in V(W_{e_j})\}$$

Since *B* is linearly ordered and *H* is without repeated hyperedge then *h* is a map. There is a bijection *g* from *B* onto *RV*, consequently $f = g \circ h$ is a map from *V* to *RV*. Let $e_i \in E$ and $v_j \in e_i$; hence $f(v_j) = \min_{l \in \{1,2,\ldots|B|\}} \{W_{e_l}, v_j \in V(W_{e_l})\} = W_{e_t}$. Because $v_j \in V(W_{e_t})$ we have $e_i \in W_{e_t}$. Let $v_q \in e_i$, $v_j \neq v_q$; $f(v_q) = \min_{l \in \{1,2,\ldots|B|\}} \{W_{e_l}, v_q \in V(W_{e_l})\} = W_{e_s}$. Because $v_q \in V(W_{e_s})$, $e_i \in W_{e_s}$. Consequently $W_{e_t} \cap W_{e_s} \neq \emptyset$ and $W_{e_t}, W_{e_s} \in A_{e_i}$. By reasoning in the same way for all vertices of e_i we can show that $f(e_i) = \{f(v_i), v_i \in e_i\} \subset A_{e_i}$. □

The presence of a morphism between the vertices of *H* and *RH* ensures that to any vertex of *H*, we can associate a vertex of *RH* in the reduction. Thereby, any partition or clustering over the vertices of *RH* can be projected onto the vertices of *H* to give rise to a partition or clustering over them.

Proposition 3. *Let $H = (V; E)$ be a simple hypergraph and $RH = (RV; RE)$ be its reduced hypergraph. We have*:

(i) $|E| = |RE|$ *if and only if for all $e \in E$, e is an isolated hyperedge.*
(ii) $|E| > |RE|$ *if and only if $|E| > 1$ and H contains a connected component with more than 2 hyperedges.*

Proof. Suppose that any hyperedge of *H* is isolated. Because *H* is simple, then for all $e_i \in E$ and for all $e_j \in E$, $i \neq j$, $e_i \cap e_j = \emptyset$. For all $e \in E$, $A_e = \{W_e\}$ hence $|E| = |RE|$.

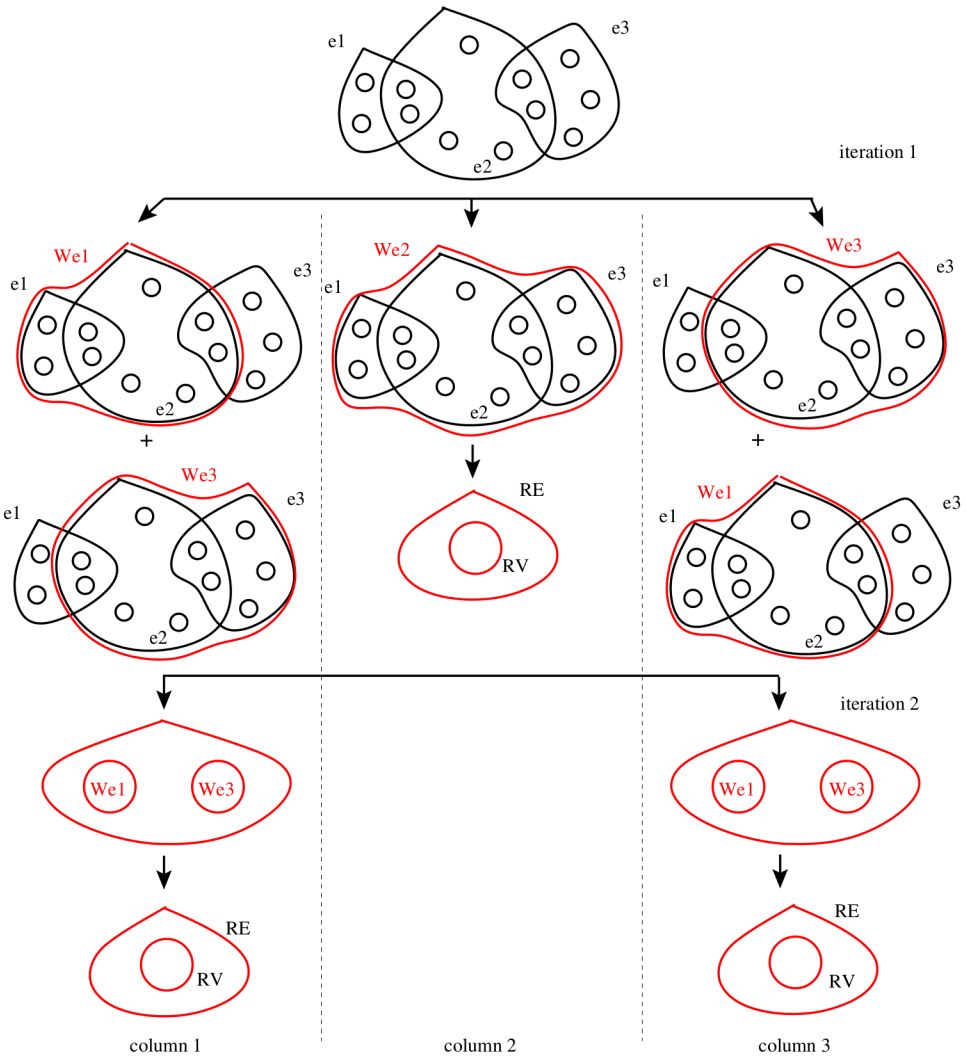

Fig. 3. Hypergraph reduction algorithm using Intersecting Hyperedges (HR-IH) with three orders. Column 1 illustrates the $\{e_1,e_2,e_3\}$ order. Column 2 illustrates the $\{e_2,e_1,e_3\}$ order. Column 3 illustrates the $\{e_3,e_2,e_1\}$ order. In the columns 1 and 3, we apply two iterations of the HR-IH algorithm.

Now assume that $|E|=|RE|$. We are going to proceed by induction on $|E|$.

If $|E|=1$ it is true; assume now that the assertion is true for any hypergraph with $|E|=m-1$, $m>1$.

Let $H=(V;E)$ be a hypergraph with $|E|=m$ such that $|E|=|RE|$. So there is a bijection between E and RE: $\forall e\in E, f(e)=A_e\in RE$; hence this bijection gives rise to a bijection between $E\backslash\{e\}$ and $RE\backslash\{f(e)\}$. By induction hypothesis the partial hypergraph $H'=(V';E\backslash\{e\})$ has all its hyperedges which are isolated. Suppose now that there is $a\in E$, $a\neq e$ such that $a\cap e\neq\emptyset$; there is a bijection from $E\backslash\{a\}$ to $RE\backslash\{f(a)\}$ and by induction hypothesis $H'=(V';E\backslash\{a\})$ has all its hyperedges isolated. Consequently there is just the hyperedge a such that $a\cap e\neq\emptyset$. So that either $W_e=\{a,e\}$ or $W_a=\{a,e\}$, hence we have $f(e)=A_e=\{W_e\}=\{W_a\}=A_a=f(a)$; hence $a=e$, contradiction.

Without losing generality, we will suppose that H is a connected hypergraph with $|E|>1$. From Proposition 1, $|B|=|V'|=|RE|$; from loop *Construction of the covering of the set of intersecting hyperedges* we cannot have $|B|<|E|$, (because H is connected with $|E|>1$); consequently $|RE|<|E|$.

Assume now that $|E|>|RE|$ then $|E|>1$, because the trivial hypergraph (without vertex) is not considered here. Moreover from (i) there is a connected component with 2 hyperedges at least. □

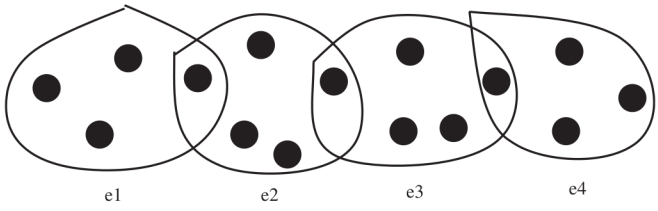

Fig. 4. The hypergraph above is connected. If we follow the given hyperedge order in Step 2 of Algorithm 1, we obtain $W_{e_1}=\{e_1,e_2\}$ and $W_{e_3}=\{e_3,e_4\}$. Consequently, in RH we have two vertices and two hyperedges $A_{e_1}=\{W_{e_1}\}$ and $A_{e_3}=\{W_{e_3}\}$ which are loops. So H is a connected hypergraph but RH is not.

Proposition 3 means that excepting the particular case where each hyperedge of H is isolated, the size of RE is always smaller than the size of E. Since the algorithm produces a neighborhood hypergraph, the same stands for the size of the vertex sets of H and RH. Consequently, the hypergraph size is effectively reduced by our algorithm.

The hyperedges set order given in Algorithm 1 can generate some problems, especially for connected components (see Fig. 4).

4.2. The proposed HR using Minimum Spanning Tree (HR-MST) algorithm

4.2.1. Basic concept

Remark 1. By using the loop defined in Algorithm 1 (Step 1) we can construct the neighborhood hypergraph of the line graph $L(H)$ of H; this hypergraph will be denoted by $H_{L(H)}$.

Using Remark 1 and in order to rectify the problem caused by the hyperedges set order, we introduce a HR-MST reduction algorithm. Both step 1 and step 2 in Algorithm 1 are replaced by the generation of a partial hypergraph H' of $H_{L(H)}$. We assume that H is connected. The generation of H' of $H_{L(H)}$ is illustrated in Algorithm 2. The reduced hypergraph $RH(RV,RE)$ has the same set of vertices than the partial hypergraph. From RH and using the W_{e_i}, we generate the set of hyperedges RE in the same way than Algorithm 1 (see Section 1).

Algorithm 2. Partial hypergraph H' of $H_{L(H)}$.

Data: $L(H)=(E;A)$ and $H_{L(H)}$
Result: Partial hypergraph H' of $H_{L(H)}$
begin
  Height function of the set of edges of $L(H)=(E;A)$;
  For $e\in E$ calculate T_e a minimum spanning tree of $L(H)$;
  The excentricity $\epsilon(e)$ is taken in T_e;
  if $\epsilon(e)=1$ **then**
    $RV \coloneqq \{w_e\}$;
    $V' \coloneqq \{e\}$;
  else
    $RV \coloneqq \{w_e\}$;
    foreach $i=1$ *to* $\epsilon(e)$ **do**
      foreach $e'\in E$, $e'\neq e$ *such that* $d(e,e')\leq 2i$ **do**
        $V' \coloneqq V'\cup\{e'\}$;
        if $d(e,e')=2i$ **then**
          $RV \coloneqq RV\cup\{w_{e'}\}$;
        end
      end
      $E \coloneqq E\backslash V'$;
    end
    if $E\neq\emptyset$ **then**
      foreach $e'\in E$ **do**
        $RV \coloneqq RV\cup\{w_{e'}\}$;
      end
    end
  end
  $H' \coloneqq (RV; RE=(a_e)_{e\in RV})$;
end

4.2.2. Basic properties

Proposition 4. *If $H=(V;E)$ is a connected hypergraph then $RH=(RV;RE)$ is also a connected hypergraph.*

Proof. Let W_{e_i}, W_{e_j} corresponding to $w_{e_i}, w_{e_j}\in RV$. We have $e_i\in W_{e_i}$ and $e_j\in W_{e_j}$. Because H is connected for $x\in e_i$ and $y\in e_j$ there is a chain from x to $y: x=v_1e_1v_2e_2\ldots v_ke_kv_{k+1}=y$. It is easy to show that $L(H)$ is connected, consequently T_e is. We have two cases:

1. $x\in e_1$ and $v_{k+1}=y\in e_k$ are on the same "branch" of T_e: by construction of Algorithm 2 there are $W_{e_u}\ni e_1$, $W_{e_v}\ni e_2,\ldots,W_{e_z}\ni e_{k+1}$, belonging to RV such that $W_{e_i}\cap W_{e_u}\neq\emptyset$, $W_{e_u}\cap W_{e_v}\neq\emptyset,\ldots W_{e_z}\cap W_{e_j}\neq\emptyset$. Consequently there is a chain from w_{e_i} to w_{e_j} in H'.
2. $x\in e_1$ and $v_{k+1}=y\in e_k$ are on two different "branches" of T_e and there is a chain from x to a vertex $u\in e$ and from u to y; in the same way than above there is a chain from w_{e_i} to w_{e_j} in H' □.

Proposition 4 shows that the connectivity problem introduced by the HR-IH algorithm has been solved by the HR-MST algorithm. It also does not involve any hyperedge order.

Remark 2. If RH is connected then H is also connected.

Proposition 5. *Let $H=(V;E)$ be a hypergraph with two hyperedges at least. $H'\subset H$ is a connected component of H if and only if there is $i\geq 1$ such the reduction of H' gives rise to an isolated hyperedge in R^iH'.*

Proof. From Proposition 3 $R^{j+1}H$ has less hyperedges than R^jH, $j\geq 0$; Moreover from Proposition 4 if R^jH is connected then $R^{j+1}H$ is connected, $j\geq 0$; the hypergraph H being finite the result follows.

Now assume that there is a $i\geq 1$ such that R^iH' has an isolated hyperedge. Hence R^iH' is connected so is $R^{i-1}H'$, by reiterating this process we show that H' is connected. □

Proposition 5 states that every connected component in a hypergraph will be reduced by an isolated hyperedge after a certain number of iterations. Since an isolated hyperedge cannot be reduced, this ensures that the recursive process will end.

Proposition 6. *Let $H_1=(V;E)$ and $H_2=(S;A)$ be two hypergraphs. If $H_1\overset{f}{\simeq}H_2$ then there is a reduction RH_2 such that $RH_1\overset{f}{\simeq}RH_2$.*

Proof. Let f be the isomorphism between H_1 and H_2. Let us reorder the set of hyperedges of H_2 in the following way: $(f(e_1)=a_1; f(e_2)=a_2;\ldots; f(e_m)=a_m)$, $a_i\in A$, $i\in\{1,2,\ldots,m\}$. Because f is an isomorphism we have

$$|a_i|=|f(e_i)|=|e_i|,\quad \forall i\in\{1,2,\ldots m\}$$

With the same notation from the algorithm

$$W_{e_i}=\{e_i=e_{k_1},e_{k_2},\ldots,e_{k_t}\}$$

$$(e_i\cap e_{k_l}\neq\emptyset,\quad \forall l\in\{1,2,\ldots t\})$$

Hence $e_i\cap e_{k_l}\neq\emptyset\Leftrightarrow f(e_i\cap e_{k_l})\neq\emptyset\Leftrightarrow f(e_i)\cap f(e_{k_l})\neq\emptyset$. So we have

$$|W_{e_i}|=|\{f(e_{k_1}),f(e_{k_2}),\ldots,f(e_{k_t}),\}|=|W_{f(e_i)}|$$

Let us $W=\{W_{e_i}, i\in\{1,\ldots m\}\}$ and $W'=\{W_{f(e_i)}, i\in\{1,2,\ldots m\}\}$. Let us $h: W\longrightarrow W'$, defined by $h(W_{e_i})=W_{f(e_i)}$. If $W_{e_i}=W_{e_j}$ then $W_{f(e_i)}=W_{f(e_j)}$, hence h is a mapping. It is surjective and injective, so it is a bijection. This bijection induces a bijection from RV to RS; we will call it g (we identify W_{e_i} with the vertex w_{e_i}). Now let A_{e_i} be a hyperedge of RH_1, then

$$A_{e_i}=\{W_{e_i}=W_{e_{l_1}},W_{e_{l_2}},\ldots W_{e_{l_k}}\}$$

with $W_{e_i}\cap W_{e_{l_j}}\neq\emptyset$ for $j\in\{1,2,\ldots k\}\}$. Hence $g(W_{e_i}\cap W_{e_{l_j}})=g(W_{e_i})\cap g(W_{e_{l_j}})\neq\emptyset$. Consequently $g(W_{e_{l_j}})=W_{f(e_{l_j})}\in A_{f(e_i)}$. Because $|A_{e_i}|=|A_{f(e_i)}|$ g is an isomorphism from RH_1 to RH_2. □

The result of Proposition 6 is important because it ensures that the reduction of two isomorph hypergraphs will also be isomorph. By extension, the reduction of a single hypergraph converges to a unique solution, modulo possible isomorphisms.

Proposition 7. *Let $H=(V;E)$ be a hypergraph and RH be its reduction. Any partition in partial subhypergraphs of $RH=(RV;RE)$ gives rise to a partition of H.*

Proof. Let $(RH_i)_{i\in\{1,2,\ldots,k\}}$ be a partition of RH in partial subhypergraphs, where $RH_i=(RV_i;RE_i)$.

Let f be the morphism defined in the proof of Proposition 2. We have

$$f^{-1}(RV)=f^{-1}\left(\bigsqcup_{i\in\{1,2,\ldots k\}}RV_i\right)=\bigsqcup_{i\in\{1,2,\ldots k\}}f^{-1}(RV_i)=\bigsqcup_{j\in\{1,2,\ldots t\}}f^{-1}(RV_j)$$

where $t=k$ when $\forall j\in\{1,2,\ldots t=k\}$ we have $f^{-1}(RV_j)\neq\emptyset$ and $t<k$ when for all $t<l\leq k$ we have $f^{-1}(RV_l)=\emptyset$. So $(H(f^{-1}(RV_i))_{i\in\{1,2,\ldots k\}}$ is a partition in partial subhypergraphs of H. □

Finally, Proposition 7 states that any clustering or partition over the vertices of RH can be projected onto the vertices of H, thanks to the morphism defined in Proposition 2.

5. Our *k*-way Reductive Hypergraph Clustering (RHC) framework

The previous section described two new hypergraph reduction algorithms. We can exploit them in a hypergraph k-clustering process based on the multilevel paradigm that consists of the following steps (see Fig. 5):

1. Set up a hypergraph $H(V;E)=R^0H(R^0V;R^0E)$.
2. Beginning with $i=0$, successively reduce $R^iH(R^iV;R^iE)$ to $R^{i+1}H(R^{i+1}V;R^{i+1}E)$ following the recursive process defined in Section 4.1.2, until a stopping condition is achieved. This procedure creates a sequence of successively smaller hypergraphs. The smallest (or coarsest) hypergraph is denoted by RH.
3. Compute an initial k-partitioning (or k-clustering) in the coarsest hypergraph RH.
4. Project the initial k-clustering of RH to the next level finer hypergraphs and use a refinement heuristic to improve the clustering, until one is available over the initial hypergraph H.

A more detailed description of these steps is given in the following sections.

5.1. Hypergraph reduction

The hypergraph to be clustered will first be reduced by our previously described hypergraph reduction processes, namely HR-IH or HR-MST. As stated by Proposition 6, the reduced hypergraphs associated respectively to two isomorph hypergraphs are also isomorph. This means that our reduction algorithm is stable and converges to a unique solution (modulo possible isomorphisms). The stopping condition will be defined by the reduction ratio $|R^iV|/|R^{i+1}V|$, where $|R^iV|$ and $|R^{i+1}V|$ denote respectively the cardinality of the vertex set of the initial and the reduced hypergraph of one iteration of the reduction algorithm (i.e. two successive coarser hypergraphs). It controls the number of reduction steps and the size of the final reduced hypergraph: when this value falls below a fixed threshold r (call it the *minimal reduction ratio*), the reduction stops to prevent from a too large amount of loss of information. In order to project the initial clustering of the coarsest hypergraph RH onto the initial hypergraph H, it is necessary to define a mapping of any vertex of H to a single vertex of RH. In the case of a generic hypergraph, such a mapping is given by the morphism defined in Proposition 2. In the special case of a neighborhood hypergraph, this mapping is straightforward because a vertex in RH corresponds to a set of hyperedges in H, and any vertex of H is associated to only one hyperedge in H (it is called the center of this hyperedge).

5.2. Initial hypergraph clustering

Given a reduced hypergraph RH, our goal is to partition the vertices v of RH into k disjoint subsets V_i $(i=0,\ldots,k)$, such that an objective function is minimized. In this paper, this function will be the *Normalized Hypergraph Cut*, as defined in [53]. Using the definitions introduced in Section 2, the optimal natural partition of

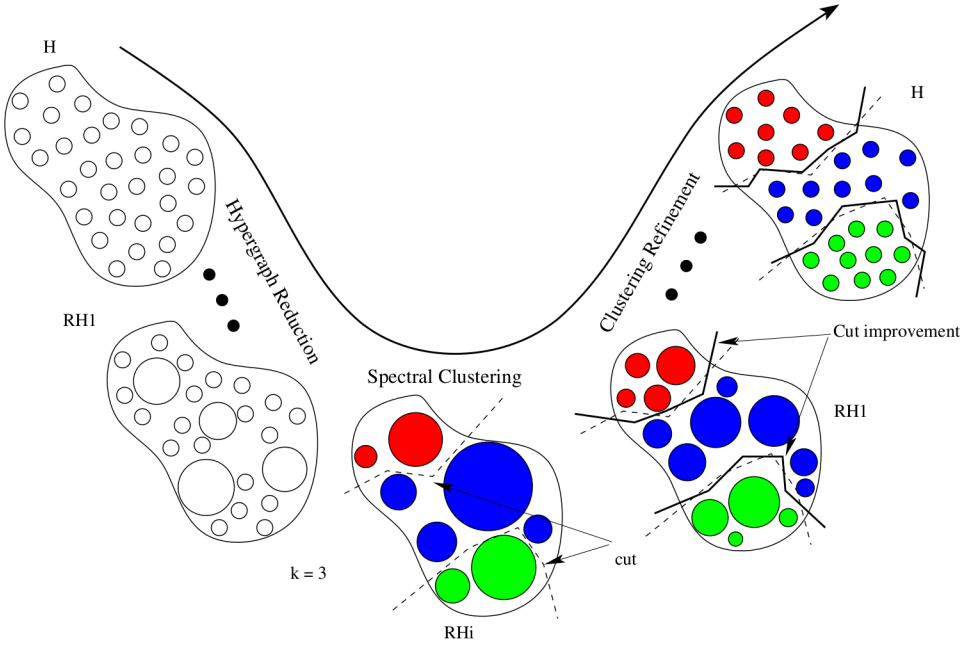

Fig. 5. RHC block diagram. After a reduction step where vertices of the initial hypergraph are merged to form levels of successively smaller hypergraphs, an initial spectral k-clustering (in this case $k=3$) is carried out over the smallest hypergraph. The clustering is then successively projected over the next level finer hypergraphs until a clustering is available over the initial hypergraph. At each level, the clustering is refined by trying to move vertices to different clusters if it improves the hypergraph normalized cut.

a hypergraph in two disjoint subsets of vertices S and S^c is given by

$$\underset{S \subset V}{\operatorname{argminc}}(S) = \operatorname{vol} \partial S \left(\frac{1}{\operatorname{vol} S} + \frac{1}{\operatorname{vol} S^c} \right) \tag{2}$$

The purpose of this function is to minimize the weights of the hyperedge that are cut, i.e. that contain vertices of different partitions, and to maximize the sum of the hyperedge weights through a same partition. Since the combinatorial problem given by Eq. (2) can be relaxed into a real-valued optimization one, we can compute the *Hypergraph Laplacian Matrix* [53] (I denotes the identity matrix)

$$L = I - D_v^{-1/2} \mathcal{H} W D_e^{-1} \mathcal{H}^T D_v^{-1/2} \tag{3}$$

The eigenvector corresponding to the second smallest eigenvalue of this matrix gives a real value for each vertex of the hypergraph, and the thresholding of these values to 0 produces an optimal bisection of the hypergraph, as shown in [53]. Thus, we can recursively bipartition a hypergraph until k sections are obtained. However, the method to compute a direct k-partitioning is preferred, because it takes advantage of the information contained in the k eigenvectors corresponding to the k smallest eigenvalues of the Laplacian matrix. Furthermore, it is computationally more efficient. These k eigenvectors provide coordinates for each vertex in a k-dimensional system. A k-clustering can be applied to these coordinates with a simple k-means algorithm [30] in order to get a k-partition. This hypergraph k-clustering algorithm is similar to the one proposed by Zhou et al. [53], and consequently our RHC framework reduces to Zhou algorithm when the hypergraph reduction step is omitted.

5.3. Clustering improvement and refinement

In this phase, the partitioning of the reduced hypergraph is successively projected to the next finer level hypergraph, and a partitioning refinement algorithm is used to optimize a given function. Most of these algorithms are based on the Fiduccia–Mattheyses (FM) [24] refinement heuristic algorithm. Proposition 7 states that a partition over a reduced hypergraph gives rise to a partition over the initial hypergraph, so such an operation can be legally applied among the different hypergraph levels. We will use in this paper the greedy refinement algorithm given by Karypis [31] that is more suitable to the refinement of a k-clustering. In this process, each vertex is visited only one time and we consider all the possible cluster migrations of this vertex. If such a migration can improve the objective function, it is applied and the other vertices are considered. In this contribution, the function to minimize will be the Hypergraph Normalized Cut given by Eq. (2).

6. Application to image segmentation

In the previous sections, we have presented a hypergraph clustering algorithm based on a novel hypergraph reduction process. Since this algorithm is generic, a practical application can be found only if one is able to build a relevant hypergraph model to represent the data of the application and a given relation among it. In this context, and to evaluate our hypergraph partitioning approach, we embedded it in a widely discussed computer vision problem: color image segmentation. In this section we show how we can model the data involved in this application by the means of hypergraphs. When such a model is available, one can exploit our hypergraph clustering algorithm to define a new image segmentation algorithm.

Let $I : V \subseteq \mathbb{Z}^2 \longrightarrow F \subseteq \mathbb{Z}^n$ be an image. Elements of V are called pixels, elements of F are called features. A distance d on V defines a grid (a connected, regular graph, without both loop and multi-edge, associated with a regular lattice $\mathbf{L}$ of $\mathbb{R}^n$). In this contribution, we will be concerned only with 8-connected grids defined by the distance $d(v,v') = \max\{|x-x'|, |y-y'|\}$, where (x,y) and (x',y') denote respectively the spatial coordinates of v and v' on the grid. Thus, we define the β-neighborhood of a pixel $v \in V$ by

$$\Gamma_\beta(v) = \{v' \in V \mid d(v,v') \leq \beta\} \tag{4}$$

Let ρ be a similarity measure on F, we have a neighborhood relation on an image defined for each pixel v by

$$\Gamma_{\lambda,\beta}(v) = \{v' \in \Gamma_\beta(v) \mid \rho(F(v),F(v')) \geq \lambda\} \tag{5}$$

Here β and λ are real values, called respectively *spatial threshold* and *feature threshold*. This neighborhood relation defines a graph on V. Recall that to each graph G we can associate its neighborhood hypergraph H_G. Consequently, to each image we can associate a hypergraph called *Image Neighborhood Hypergraph* (INH) [6]

$$H_{\Gamma_{\lambda,\beta}} = (V; (\{v\} \cup \Gamma_{\lambda,\beta}(v))_{v \in V}) \tag{6}$$

The spatial and color neighborhood $\Gamma^{color}_{\lambda^c,\beta}$ is generated in following Eq. (5). This requires the definition of a distinct similarity measure for color feature. Throughout this paper, all the similarity measures will be normalized by a Gaussian kernel $\rho^F(F(v),F(v')) = \exp(-d^F(F(v),F(v'))/\sigma^F)$, where d^F is a distance measure over F, and σ^F is computed as the standard deviation of d^F over 200 random pairs of vertices in V. We define the color distance d^c as the Euclidean distance in Lab color space [47]. In the case of grayscale images, $d^c(v,v') = |I(v)-I(v')|$, where $I(v)$ denotes the gray level of pixel v.

7. Simulations and discussions

7.1. Experimental protocol

Graph and hypergraph clustering algorithms are difficult to evaluate since many of existing approaches are reliable in a certain context and are generally designed for a particular application. In fact, clustering algorithms are mainly evaluated and compared to others throughout this application, with no indication if the same approach may be effective when transposed to another application, or for generic graph or hypergraph clustering purposes. For this reason, and in order to assess the effectiveness of the proposed RHC approach, we first evaluate its performances in a generic hypergraph clustering problem, and then in the context of color image segmentation. In the following, the minimal reduction ratio r involved in our algorithm has been fixed to 1.5 after some experiments. In HR-MST, the weight of the edges of the line graph between the hyperedges e_1 and e_2 is given by $w(\{e_1,e_2\}) = 1/|e_1 \cap e_2|$ (see Fig. 6). The reason for this choice comes from an intuitive idea that two hyperedges sharing a large number of vertices are more likely to connect vertices from a single cluster in the hypergraph, and thus may be linked by an edge associated with a small weight (recall that the objective of the MST computation is to minimize the total edge weight over the whole tree). In the following, we work exclusively with unweighted hypergraphs (i.e. $\forall e \in E, w(e) = 1$). All the experiments were performed on a machine with the following characteristics: Intel Xeon 2.67 GHz, 4 GB RAM.

7.2. Evaluation of hypergraph clustering

The simplest way to judge the quality of a hypergraph clustering algorithm is to compare the partition obtained on a

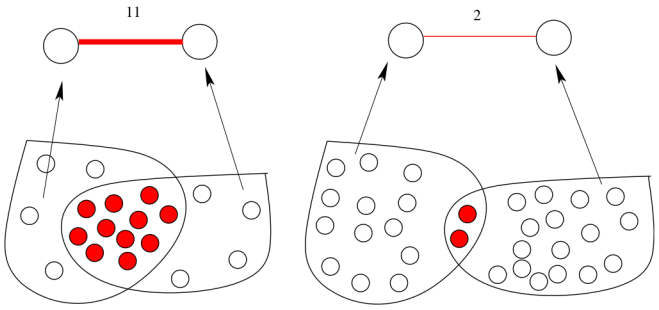

Fig. 6. An edge of the line graph (connecting two hyperedges) will be weighted by the number of vertices shared by the corresponding hyperedges. The weight is then converted to a distance by taking its multiplicative inverse.

hypergraph where the cluster structure is known to the ground truth partition. For the sake of generality, such a hypergraph must be randomly generated to ensure that the algorithm is not specific to a given application or hypergraph structure. Unfortunately, such a generative model does not exist at that time in the case of hypergraphs. But recall that from every arbitrary graph, we can build a neighborhood hypergraph. Hence, gathering a hypergraph clustering benchmark becomes straightforward because several computer-generated graph algorithms have been introduced for this purpose. In particular, the *planted* ℓ-partition *model* [17] has become quite popular in the last years. A special case of the planted partition model designed by Girvan and Newman [27] has also gained a standard status and is a widely used class of graphs in the graph clustering community. Such a graph is defined by ℓ equally sized clusters of g vertices. Each vertex is randomly linked by an edge to another vertex with a probability p_{in} if the two vertex belong to the same group, and a probability p_{out} otherwise. Those properties lead to graphs where each vertex has approximatively the same degree, which is not representative of real-world problems. Recently, Lancichinetti et al. [34] designed an algorithm to generate graphs based over the planted partition model. In particular, the cluster sizes and the vertex degrees are chosen randomly from a power law distribution. The complexity of the clustering solution can be easily given by the *mixing parameter* λ_t, which represents the fraction of the incident edges of a vertex that connects it to another vertex of a different cluster. We consider that the graph has a community structure when $\lambda_t < 0.5$. This class of graphs (called the LFR benchmark) is more adequate to represent real-world problems, and consequently we will use it to evaluate our hypergraph clustering algorithm. Note that the exploitation of a graph benchmark is also interesting because it allows us to directly compare graph-based and hypergraph-based methods.

Each graph generated by this method is then converted to a neighborhood hypergraph with the definition given in Section 2. The performance of a clustering algorithm is then evaluated by comparing the partition obtained by the algorithm and the ground truth planted partition of the hypergraph. Such a comparison measure is the normalized mutual information (I_{norm}) [19] that represents the amount of information shared by the two partitions and serves as a similarity measure. Values are comprised between 0 and 1, when the value 1 indicates that the two provided partitions are identical. The graphs used in these experiments are built following three parameters: the number of vertices N, the number of clusters k and the mixing parameter λ_t that adapts the complexity of the solution. For each set of parameters, a certain number of graphs (5 in our experiments) are generated and different clustering algorithms are applied over the resulting neighborhood hypergraphs. The I_{norm} values associated with the partitions obtained with a single algorithm are then averaged over all the hypergraphs to provide one value per couple algorithm/set of parameters. Different algorithms have been compared:

- Zhou [53]: Zhou's standard spectral hypergraph clustering algorithm.
- Hmetis [31]: a hypergraph partitioning tool based on the multilevel paradigm. It is still widely considered as the best existing software in VLSI domain [3].
- FC, EC, MHC: our RHC framework but using standard hypergraph reduction algorithms typically used for hypergraph partitioning [31] (see Section 3).
- HR-MST: our proposed RHC algorithm, using our proposed HR-MST hypergraph reduction approach.
- HR-IH: our proposed RHC algorithm, using our proposed HR-IH hypergraph reduction approach.
- Graph: the typical Shi and Malik's spectral graph partitioning algorithm [46]. Yu et al. direct k-clustering version [51] was used along these experiments.

The considered algorithms only need in general to set the number of clusters k, except for Hmetis that also take two additional parameters: the reduction algorithm to use in its multilevel implementation, and the *balance parameter* (an integer greater than 5). The purpose of the latter is to control the relative sizes of the different clusters, for instance specifying a value of b means that the size of a single partition should not be more than b% greater than the average size. FC will be used as the reduction algorithm (as advised in [31]). As the variation of the balance parameter did not influence the results of Hmetis in preliminar experiments, it will be set at a large value (typically 10 000) in order to allow clusters of variable size.

In order to evaluate the behavior of the different algorithms according to the nature of the data, one parameter of the graph generation process is varied at a time. The two other parameters are fixed at standard values $N=10\,000$, $k=10$ and $\lambda_t = 0.2$. The values of 10 000 and 10 have been chosen after some experiments that suggested that those values allow to get the most stable clustering results among different graphs generated with the same parameters. The value of 0.2 for λ_t is suggested as a standard value by the designers of the LFR benchmark [34]. For more clarity, the results are split between those obtained with the RHC framework using different hypergraph reduction algorithms, and a comparison between our RHC framework (using HR-MST) and the other clustering softwares mentioned above.

We will start by examining the influence of the λ_t parameter. From the curves in Fig. 7(a), we can see that the proposed HR-MST algorithm provides the best partitions within the RHC framework, because its curve is always above the curves that correspond to the other hypergraph reduction algorithms. In particular, HR-MST is better than the FC algorithm, which is widely considered as the most efficient in VLSI [31] and image segmentation [21] domains. This result demonstrates that it is also the case with application to generic data. The weak performances of the HR-IH and MHC algorithms can be explained by the fact that when the λ_t parameter becomes high ($\lambda_t > 0.3$), two vertices from different clusters are more likely to belong to the same hyperedge, and the clustering results are biased because HR-IH and MHC tend to merge vertices that are found in the same hyperedges. On the contrary, the other algorithms take care of the connectivity between the vertices and are more prone to merge vertices that are strongly connected (i.e. connected with a large number of hyperedges), reducing the clustering errors. In particular, the computation of the MST of the line graph has the ability to neglect the pairs of hyperedges that share a small number of vertices. Finally, two hyperedges order are considered in the HR-

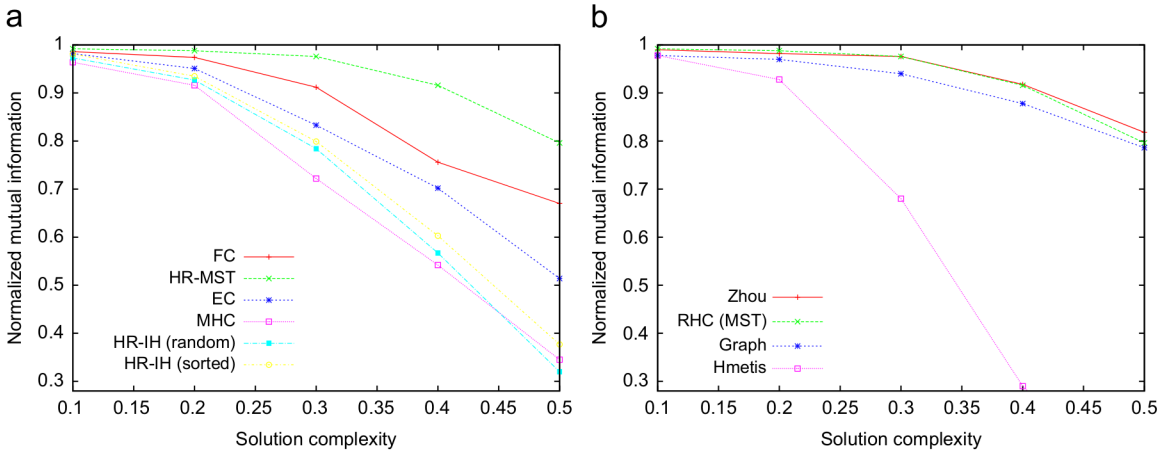

Fig. 7. Normalized mutual information of hypergraph partitions obtained with (a) the RHC framework using different hypergraph reduction algorithms and (b) different graph/hypergraph clustering softwares, according to the complexity of the solution (given by the mixing parameter λ_t).

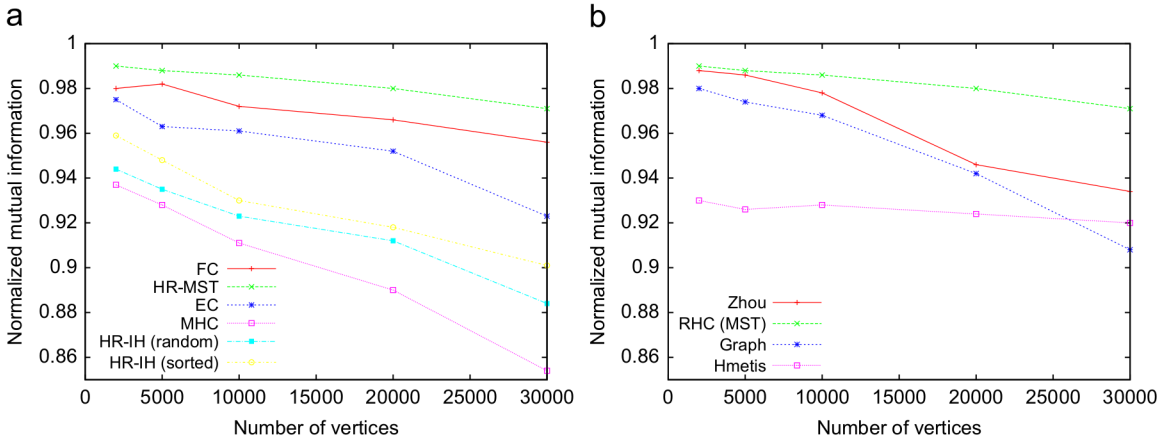

Fig. 8. Normalized mutual information of hypergraph partitions obtained with (a) the RHC framework using different hypergraph reduction algorithms and (b) different graph/hypergraph clustering softwares, according to the number of vertices N of the hypergraphs.

IH algorithm: random and sorted (in this case, hyperedges are visited by descending weights, then size). The results show that despite the connectivity problem introduced by the algorithm (see Section 4.1.3), the hyperedge order has a small influence over the clustering results. Although, the sorted order allows to get a slightly better performance. It is also important to note that the results for HR-IH remain strongly relevant when the data is well correlated (i.e. $\lambda_t < 0.3$). In conclusion, we should note that the HR-MST algorithm does not need the user to define a hyperedge order.

We will now compare the results of our RHC framework (using HR-MST) with other graph/hypergraph clustering software. By looking at the curves of Fig. 7(b), we can see that despite the implicit loss of information introduced by the hypergraph reduction, the solution found by our HR-MST algorithm is often very close to the solution provided by Zhou's direct spectral hypergraph clustering method (without a hypergraph reduction step), and outperforms the other algorithms. Furthermore, in many cases the solution is better when we consider hypergraphs with a clear community structure ($\lambda_t < 0.4$). It is also interesting to note that the Hmetis algorithm, based only on heuristic methods, fails to detect the community structure when the solution complexity becomes high. This can be explained by the random behavior of those techniques, especially in the initial partitioning phase where a randomly computed initial partition is not sufficient to obtain a good clustering. In addition, we can observe that the curves associated with the hypergraph-based algorithms (excluding the particular case of Hmetis) are in general always above the curve representing the performance of the spectral graph-based clustering approach, despite the fact that it is still regarded as the best direct graph k-clustering algorithm. This shows particularly the advantage of using a hypergraph-based method which exploits a richer model of data representation.

If the spectral-based graph/hypergraph clustering methods are able in general to provide the best partitioning results, this is not always verified when the size of the data becomes high and difficult to handle. This result is shown in Fig. 8(b), that represents the I_{norm} values obtained when the number of vertices in the hypergraph varies ($k=10$ and $\lambda_t = 0.2$). One can observe that the performance of the spectral methods gradually decrease when the size of the data (i.e. the number of vertices to consider) increases. On the contrary, the other algorithms (based on the multilevel paradigm with a reduction step) are less prone to this phenomenon. This result demonstrates that a hypergraph reduction step, associated with an initial spectral clustering and a refinement step, becomes very effective relatively to a direct spectral clustering when the size of the problem becomes high. In particular, our proposed method exploiting our HR-MST algorithm achieved the best performance (see Fig. 8(a) for a comparison between the reduction algorithms). This result indicates that a multilevel

approach can be very efficient for large datasets, such as image or video segmentation when the number of elements involved cannot be handled by a typical spectral clustering algorithm.

To illustrate this statement, the Hmetis algorithm has been shown very effective for partitioning hypergraphs obtained from real data, particularly in VLSI design [31] and image segmentation [42] with the INH model. In fact, spectral algorithms achieve in general the best results for generic hypergraph clustering purposes (as demonstrated above) and small datasets in application to categorical data classification [53]. However, Fig. 9 demonstrates that Zhou's spectral algorithm is not appropriate in the case of image segmentation via an INH model, while Hmetis and our proposed RHC algorithms achieve reliable results. In addition, the complexity of a spectral algorithm becomes problematic in application to real data. In Fig. 9, the size of the original image has been reduced by a factor of 2 in order to compute a segmentation with Zhou algorithm in a reasonable amount of time. Despite this rescaling, the segmentation using Zhou algorithm takes 29.38 s, compared to 0.96 s with our proposed approach (using the HR-MST reduction process) and 0.64 s with Hmetis.

Fig. 10 presents the I_{norm} values obtained when the number of clusters k varies. We can remark that the different algorithms share the same behavior: the quality of the clustering decreases when k increases, which is a coherent result when we suppose that a solution with a large number of clusters is harder to find than a solution with a smaller number of clusters. In fact, we did not observe specific differences of behavior between the different algorithms for this parameter.

Finally, Fig. 11 presents the computation times obtained with different algorithms when N and k vary. First, it is important to note that Zhou algorithm (results for Shi and Malik graph algorithm are not presented because they are very close to those obtained with Zhou algorithm) is the one that comes along with the heaviest computational burden. In particular, its complexity depends highly on the number of vertices N (it seems to be the case at a lower scale for the other algorithms) and the number of clusters k. The latter result does not stand for the algorithms that adopt a multilevel approach, for which computation times are very stationary. The important thing to note is that the HR-IH algorithm is by far the fastest algorithm, and that HR-MST, in addition of being the algorithm that provides the best clustering results, displays reasonable computation times, despite the explicit building of the MST and the line graph of the hypergraph. In conclusion, one should prefer to use our HR-IH algorithm in a problem where the main objective is to speed up the clustering (if it involves datasets of considerable size, like high-definition images for instance). The HR-IH algorithm have already shown very promising results in an application for image superpixels generation [7] showing that it can be very efficient when a relevant hypergraph representation of the date is exploited. The HR-MST algorithm seems to be a good compromise between the quality of the clustering and the algorithm complexity.

Fig. 9. (a) Original color image (321 × 481 pixels) from the Berkeley Segmentation Database [36], and segmentation results for (b) Zhou algorithm, (c) Hmetis algorithm and (d) our proposed RHC (using the HR-MST reduction process) algorithm (with the number of clusters $k=8$). Segmentation results are obtained by partitioning an INH built with parameters $\lambda = 0.85$ and $\beta = 1$ (see Section 6). The segments boundaries are displayed in red and superimposed to the original image. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

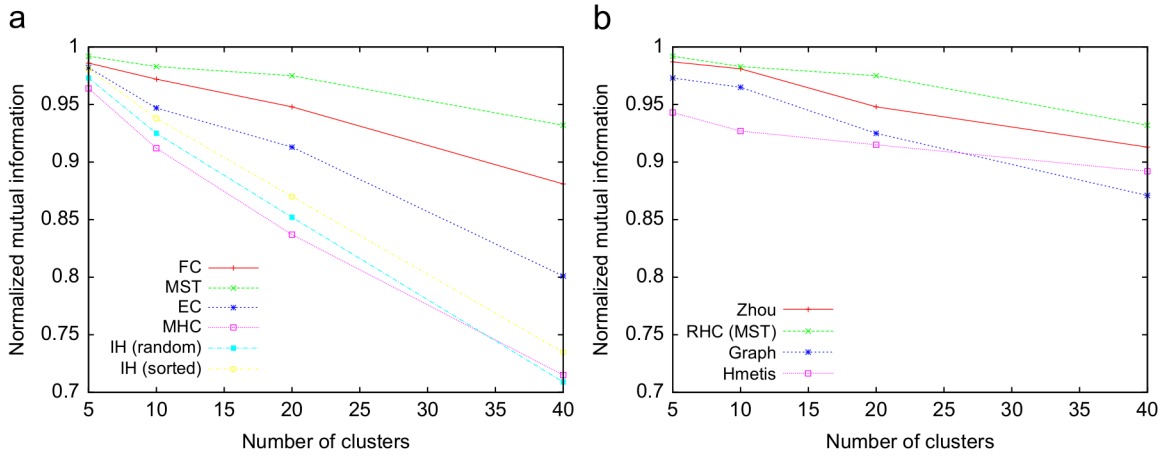

Fig. 10. Normalized mutual information of hypergraph partitions obtained with (a) the RHC framework using different hypergraph reduction algorithms and (b) different graph/hypergraph clustering softwares, according to the number of clusters k.

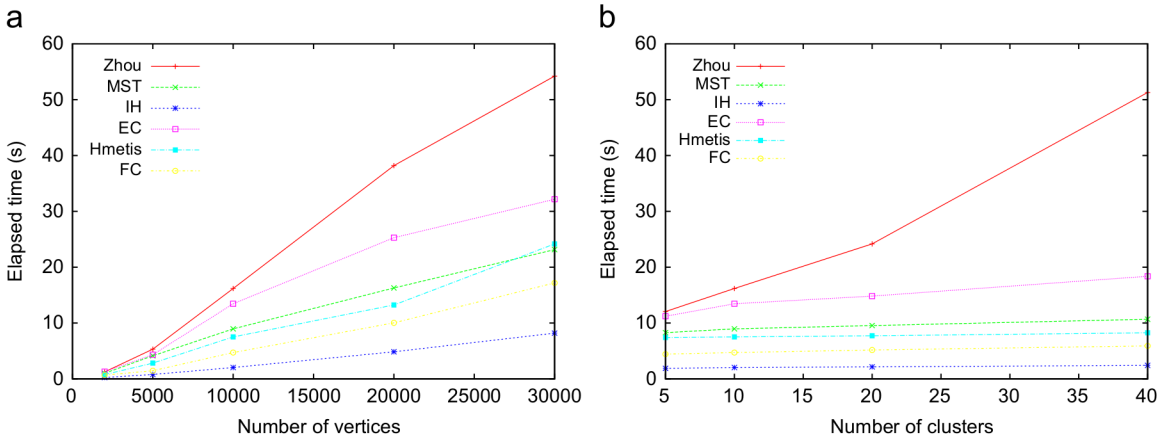

Fig. 11. Computation times for different clustering algorithms according to (a) the number of vertices N in the hypergraphs ($\lambda_t = 0.2$, $k = 10$) and (b) the number of clusters k ($\lambda_t = 0.2$, $N = 10\,000$).

7.3. Results of color image segmentation

We will now evaluate our hypergraph clustering algorithm using HR-MST (namely RHC) in the field of color image segmentation. Experiments have been carried out over a subset of 30 images of the well-known Berkeley Segmentation DataBase (BSDB) [36], that provides a set of nature scenes images with human ground-truth segmentations. We limited our experiments to 30 images, because the INH representation involved in our segmentation algorithm (see Section 5) is at that time not able to handle texture. It is also the case for the other algorithms that are compared in this section. As a consequence, only images from the BSDB that display little texture have been selected. The INH model is generated with the standard spatial threshold value $\beta = 1$, which has been shown to have a small influence on the segmentation results [21].

We will give some comparison results with three other image segmentation algorithms:

- *NCut*: it uses a graph representation of images and is based upon Shi and Malik's NCut framework [46]. The segmentation is treated as a graph partitioning problem, solved by the computation of eigenvectors of a normalized graph Laplacian matrix. This computation is speeded up by working on a multiscale decomposition of the graph [18]. This implementation involves only color features in the computation of the edge weights (as well as in our INH model).
- *Hmetis*: This algorithm [42] exploits a hypergraph representation of images, and attempts to find a partitioning of the hypergraph to get a segmentation, based on the heuristic multilevel hypergraph partitioning framework introduced in [31]. The hypergraph partitioning is provided by the hMETIS package [31].
- *Mean Shift*: The standard Mean Shift image segmentation algorithm [16]. We will use a speeded up version of the algorithm [39].

The idea behind the choice of the two first approaches is to demonstrate the advantages of a hypergraph-based image representation compared to a graph-based one (NCut), despite the fact that the NCut algorithm computes a k-partitioning of a graph that is close to the optimum. In addition, we wanted to compare the ability of our hypergraph clustering approach based on the combination of a hypergraph reduction algorithm and spectral hypergraph partitioning techniques, to an approach that is exclusively based on heuristic methods (Hmetis). For this purpose, it is important to use the same hypergraph representation in our proposed (RHC using HR-MST) and Hmetis frameworks to judge the quality of the different hypergraph partitioning algorithms according to a fixed set of parameters. The use of Hmetis is also motivated by the fact that it has been found particularly meaningful in the context of image segmentation [42], and that a direct spectral hypergraph partitioning approach such as Zhou algorithm [53] is not appropriate within this application (see Section 7.2). Finally, the results are compared to those provided by Mean Shift, a well-known and standard segmentation algorithm in the computer vision domain, in order to judge the reliability of our hypergraph clustering algorithm in this field.

NCut only needs the user to set up the number of segments k. For each algorithm (excluding the MS approach that automatically computes the number of segments), different values of the number of segments k were tested (from 2 to 20). In addition to this parameter, our algorithm also requires the color threshold λ in order to build the INH (see Section 6). It is also true for Hmetis that uses the same hypergraph representation. Discussions about the influence of this parameter in the segmentation results can be found in [8,21,42]. In these experiments, λ will be computed in an adaptive way at each pixel, $\lambda(v) = (\overline{\|F(v')\|} / \sigma(\|F(v')\|))_{v' \in \Gamma_\beta(v)}$ designing the color threshold at pixel v, while $\overline{\|F(v')\|}$ and $\sigma(\|F(v')\|)$ design respectively the mean and the standard deviation of the magnitudes of the color vectors of each pixel (including v) in the β-neighborhood of v (see Section 6). As in Section 7.2, the reduction approach adopted for Hmetis will be the FC algorithm, and the balance parameter will be set to a large value (typically 10 000) in order to allow the presence of segments of different sizes in the images. Two algorithms specify the behavior of the Mean Shift algorithm, namely the spatial and color bandwidth parameters that control the precision of the segment detection for the color and spatial features respectively (see [16] for more details). As suggested in [39], we tested values from ranges [5,10] for the color bandwidth and [4,256] for the spatial bandwidth. Best results occur in general for the smallest values in the specified ranges. Since our goal was not to discuss about the relative influence of each parameter, the parameters of the different algorithms were set in order to capture the best segmentation after some experiments, which provided a result for each couple image/algorithm. All algorithm results are compared with human boundary maps obtained from the Berkeley image segmentation database [36]. For quantitative comparison, the performance of the four considered algorithms was analyzed in terms of precision and recall [41]. The precision and recall measures are particularly meaningful in the context of boundary detection when we consider applications that make use of boundary maps,

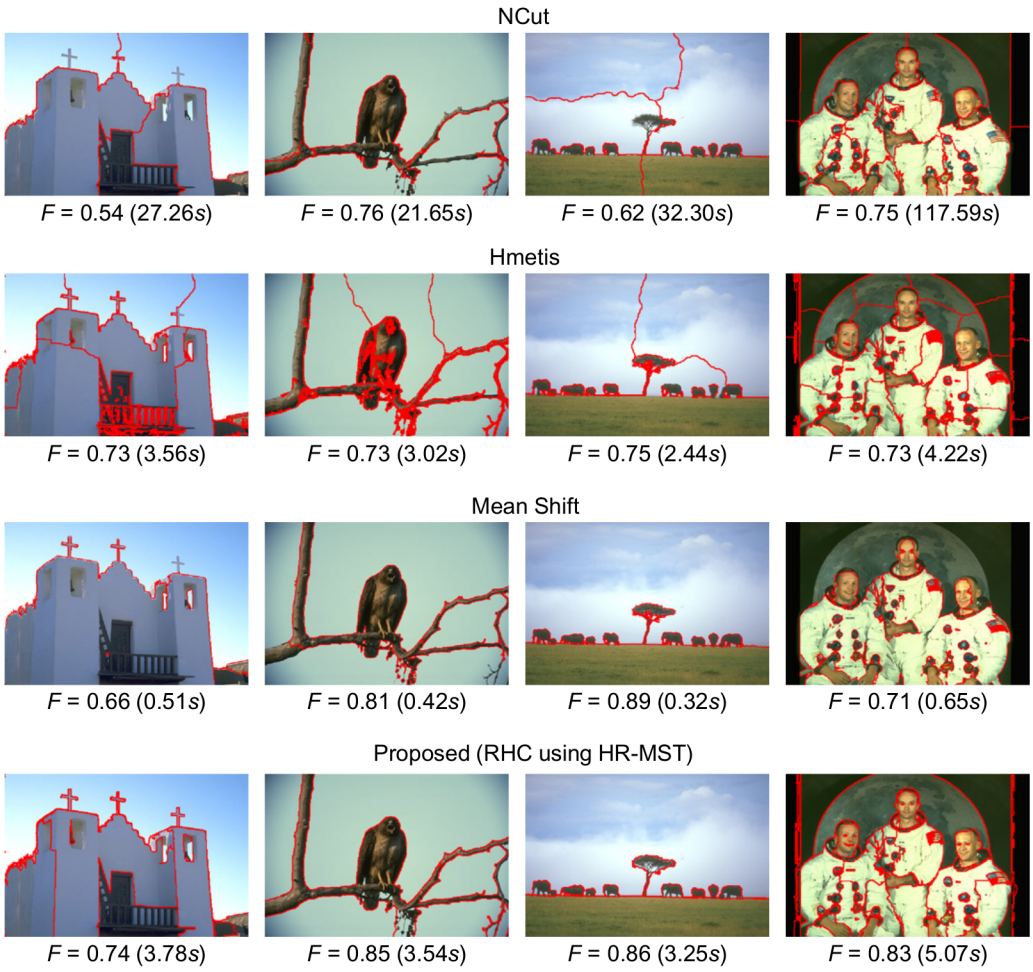

Fig. 12. Results of segmentations of images from the BSDB obtained with NCut [18], Hmetis [42], Mean Shift [39] and our proposed algorithm (using HR-MST). *F*-scores and computation times are displayed for each result.

such as stereo or object recognition. It is reasonable to characterize higher level processing in terms of how much true signal is required to succeed R (recall), and how much noise can be tolerated P (precision). A particular application can define a relative cost α between these quantities, which focus attention at a specific point on the precision-recall curve. The *F*-score [41], defined as $F = PR/\alpha R + (1\alpha P)$ captures this trade-off as the weighted harmonic mean of P and R. The *F*-score is valued between 0 and 1, where larger values are more desirable. We set α to 0.5 in our experiments as in [37]. It is important to note that all *F*-scores displayed in this paper are automatically computed using the benchmark software available on Berkeley's website.[1]

Fig. 12 shows some examples of segmentation results obtained on images from the BSDB subset with RHC (using HR-MST), NCut, Hmetis and Mean Shift algorithms, along with their associated *F*-scores and computation time. The segment boundaries are colored in red and superimposed to the original color image. Average *F*-scores and computation times obtained with the different algorithms over the whole 30 images subset of the BSDB are reported in Table 1. We can observe from these values that our segmentation algorithm displays the highest evaluation measures, and consequently outperforms the others. In particular, a hypergraph representation tends to detect more accurately the boundaries of the image, as the NCut approach often fails at detecting small objects

Table 1
Average *F*-score values (standard deviation is also displayed) and computation times obtained from different image segmentation algorithms over the BSDB subset.

Algorithm	*F*-score	Time (s)
Proposed (RHC using HR-MST)	0.74 ± 0.117	4.31
Mean Shift	0.69 ± 0.147	0.54
Hmetis	0.68 ± 0.099	3.54
NCut	0.64 ± 0.115	44.28

(take for instance the animals in the image of the third column in Fig. 12). This can be explained by the fact that representations involving multi-dimensional relationships (by the means of hyperedges) between elements is closer to the human visual grouping system than a graph-based representation that simply takes into account pairwise relationships. In addition, our proposed hypergraph clustering algorithm gives better results than the Hmetis algorithm, even when we consider that exactly the same hypergraph representation is used in both cases. In particular, the Hmetis algorithm has a tendency to isolate small regions in the image (resulting in thick boundaries). This is due to the presence of small connected components in the INH, and can be explained by the use of local thresholding in hyperedges generation. This phenomenon appears in the noisy or textured images, showing that the proposed hypergraph representation is not suitable for such images. Our algorithm also performs well compared to the Mean Shift algorithm,

[1] 〈http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/〉