Combining Semi-supervision and *Hubness* to Enhance High-dimensional Data Clustering

Mateus C. de Lima, Maria Camila N. Barioni, Humberto L. Razente

Faculdade de Computação (FACOM)
Universidade Federal de Uberlândia (UFU), Uberlândia, MG, Brazil
{mateuscurcino, camila.barioni, humberto.razente}@ufu.br

Abstract. The *curse of dimensionality* turns the high-dimensional data analysis a challenging task for data clustering techniques. Recent works have efficiently employed an aspect inherent to high-dimensional data in the proposal of clustering approaches guided by *hubs* which provide information about the distribution of the data instances among the *K-nearest neighbors*. However, *hubs* are not enough to reflect the implicit data semantics, allowing to incorporate other strategies to deal with this issue. In order to cope with both issues, the high dimensionality of the data and the achievement of meaningful clusters, this article presents a clustering approach that explores the combination of two strategies: semi-supervision and density estimation based on *hubness scores*. The experimental results conducted with 23 real datasets show that the proposed approach has a superior performance when compared with the *HPKM*, the *Kernel k-means*, the *DBSCAN* and the *SSDBSCAN* algorithms.

Categories and Subject Descriptors: H.2.8 [**Database Applications**]: Data mining; H.3.3 [**Information Storage and Retrieval**]: Clustering

Keywords: Data Mining, High-dimensional Data Analysis, *Hubness*, Semi-Supervised Clustering.

1. INTRODUCTION

There exist various application domains for which the employment of data mining techniques is useful such where the dimensionality of the data is notably elevated. Among such are databases where each data instance is described by a collection of features, as in the case of images and gene expressions. The so-called *curse of dimensionality* [Samet 2006] makes data analysis in higher dimensions a challenge for any data mining technique based on distance calculations, since the extent to which the dimension increases, the sparseness of data also increases, and so does the difficulty to differentiate data instances. In this context, it is interesting to consider the employment of techniques that allow for the mitigation of these prejudicial effects on the effectiveness and efficiency of data mining algorithms.

Traditionally, the question of higher dimensions has been dealt with in the scientific literature through the use of strategies that look to mitigate the effects of the *curse of dimensionality* by means of the extraction and the selection of features or through the proposal of methods that work in subspaces. However, information can be lost when the dimensionality is reduced [Samet 2006]. In an opposing manner, more recent strategies have employed in an efficient manner an aspect inherent to high-dimensional data in the proposal of techniques that allow for the classification [Tomasev and Mladenic 2013][Buza 2016], the search for the *nearest neighbors* [Flexer and Schnitzer 2013][Tomasev and Mladenic 2014] and the clustering [Tomasev et al. 2014][Tomasev et al. 2015] in high-dimensionality databases. This aspect, denominated *hubness*, consists of the tendency that some data

This work has been supported by CNPq (Brazilian National Council for Supporting Research), by CAPES (Brazilian Coordination for Improvement of Higher Level Personnel) and by FAPEMIG (Minas Gerais State Research Foundation).

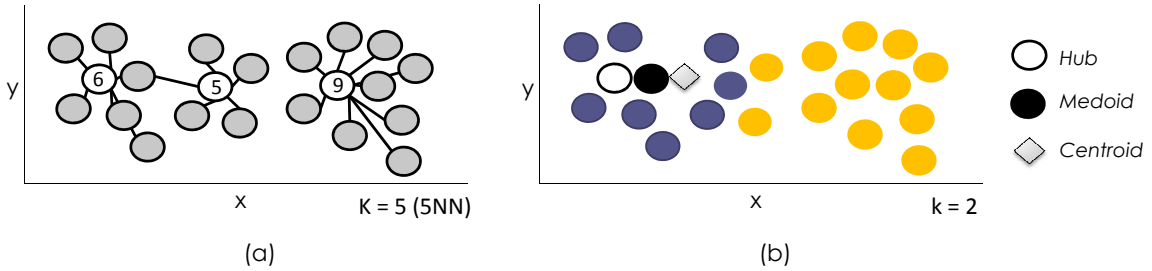

Fig. 1. The use of *hubs* as cluster prototypes. (a) The *Hubs* and respective scores. (b) *Hubs* and traditional cluster prototypes.

instances, called *hubs*, occur with greater frequency in the list of K*-nearest neighbors* than other instances, for a given K.

Considering the clustering task, previous works employed *hubness* information to determine the prototypes of each data partition leading to better clustering solutions for high-dimensional data. For example, Figure 1(a) illustrates a 2D dataset where the white circles indicate the data instances that occur in the list of the *5-nearest neighbors* of the other data instances. The number placed inside the circles represent the number of times that the data instance occurs in the list of the *5-nearest neighbors* of other data instances. Figure 1(b) illustrates the existing correlation between *hubness* and the proximity to traditional cluster prototypes for partitional clustering approaches.

Although clustering techniques directed by the *hubness score* have presented good results when dealing with high-dimensional data, it is of importance to highlight that as the *hubs* are centers of influence, possible inaccuracies associated with them can be easily propagated [Tomasev and Mladenic 2013]. A particular strategy that can be used in order to minimize the risk of inducing an unsuitable partitioning in the data, since the *hubs* may not reflect the implicit data semantics (the semantic gap between the metric and the data), is the incorporation of semi-supervision [Basu et al. 2008].

This article presents a new clustering approach that explores the combination of semi-supervision strategies and the use of *hubness score* in respect to the instances of data, with the focus being high-dimensional data. This approach revisited the unsupervised algorithm *HPKM* [Tomasev et al. 2011], which gave rise to the method denominated as *SSHub Clustering*. Although a preliminary version of the work described herein was presented in a previous short paper [Lima et al. 2016], new aspects are dealt with in this article. The present article integrates the concepts introduced in the previous paper and goes beyond these through the presentation of a more detailed description of a new and improved version of the proposed algorithm and the results obtained through an exhaustive set of experiments that used 23 sets of real data. In addition to the *HPKM*, the *Kernel K-means* [Dhillon et al. 2004] and the *DBSCAN* [Sander et al. 1998], the *SSDBSCAN* [Li et al. 2014] algorithm was also selected as a baseline for comparison as a representative of the semi-supervised clustering algorithms. Moreover, additional experiments varying the parameters of the algorithm were included. By analyzing the obtained experimental results, it is possible to state that the *SSHub Clustering* achieved better performance when applied to real datasets of different sizes, which surpassed the effectiveness of the four selected algorithms used on a comparison basis.

The remainder of the article is organized as detailed in the following. Section 2 presents the fundamental concepts and correlated research. The *SSHub Clustering* is described in Section 3. The discussion regarding the obtained experimental results is presented in Section 4. Finally, the conclusions and future work are described in Section 5.

2. BACKGROUND AND RELATED WORK

This section presents the fundamental concepts for the presentation of the proposed *SSHub Clustering* algorithm. The symbols and definitions described in Table I are used throughout this article.

Table I. Summary of Symbols and Definitions.

Symbols	Definitions
k	number of clusters
$C = \{c_1; ...; c_k\}$	partition set
$X = \{x_1; ...; x_n\}$	set of data instances to be clustered
K	neighborhood cardinality of each instance (K-*nearest neighbors*)
$h_K(x)$	*hubness score* of the instance x considering the neighborhood K
$N = \{h_K(x_1); ...; h_K(x_n)\}$	set of data instances *hubness scores*
p_i	main representative of cluster c_i
a_i	auxiliary representative of cluster c_i
A_i	set of auxiliaries representatives of cluster c_i
Q_i	set of representatives (main and auxiliaries) of cluster c_i
$r_{ml}(x_i, x_j)$	*must-link* constraint between x_i and x_j
$r_{cl}(x_i, x_j)$	*cannot-link* constraint between x_i and x_j
R_m	set of r_{ml}
R_c	set of r_{cl}
R	$R_{ml} \cup R_{cl}$

In a general sense, the data-clustering task aims to find clusters according to a similarity measure, in a manner that the data instances from a related cluster possess high similarity, while the data instances from different clusters possess low similarity [Aggarwal and Reddy 2013]. To compute this measure it is necessary to have a description of the data instances and a distance function. Thus, depending on the data domain, the instances can be described by a set of attributes of traditional domains (for example, text or numbers) or by a collection of pre-defined feature descriptors inherent to the data (considering the image domain, for example: color, texture and shape, among other features). The choice of a distance function to be employed also depends on the deployed data domain, being that among the most commonly used are the distance functions from the *Minkowski* family [Taniar and Iwan 2011].

It is important to highlight here that the expressiveness of this similarity measure is an important factor for obtaining good results in data clustering processes. When dealing with high-dimensional data this desirable feature of the similarity measure degrades due to the effects of the *curse of dimensionality*, thereby reducing the capacity of the similarity measure in differentiating pairs of data elements. In an attempt to mitigate the undesirable effects of this curse in high-dimensional data clustering processes, two approaches may be employed: 1) apply techniques to reduce the dimensionality as a first step before performing data clustering, or 2) use specialized solutions for analyzing high-dimensional data.

The strategies proposed in the scientific literature from this area for dimensionality reduction arise from research areas such as Pattern Recognition, Statistics and Information Theory, which aim at reducing the number of dimensions while excluding irrelevant, low relevance or redundant features from the description of the data instances. This strategy can be divided into two categories: 1) feature selection, and 2) feature extraction.

The techniques based on the approach of feature selection have as their objectives to choose a small subset of features according to a given criterion. Here are some examples of feature selection techniques: techniques based on the calculation of the *Fractal Dimension* [Traina-Jr et al. 2010], *Information Gain* [Kullback and Leibler 1951] and *Relief* [Kira and Rendell 1992], among others. On the other hand, those techniques based on the feature extraction approach map the original data space in a lower dimensional space. Among the feature extraction techniques proposed one can cite, *Principal Component Analysis* (PCA) [Hair Jr. et al. 1995], *Multidimensional Scaling* (MDS) [Borg and Groenen 2005] and *FastMap* [Faloutsos and Lin 1995].

The techniques for dimensionality reduction, in general, provide only a subspace of original data, in which the data clustering process can be performed. However, for situations where different features

can be relevant to different groups in one singular data clustering, such methods tend to fail. In these cases, algorithms for subspace clustering appear as a new strategy for analyzing high-dimensional data [Aggarwal and Yu 2000][Kailing et al. 2003][Kailing and H.P. Kriegel 2004][Muller et al. 2009].

Another example of a specialized solution for the analysis of high-dimensional data are the clustering algorithms based on *hubness* [Tomasev et al. 2011]. Differently from the previous approaches, instead of trying to avoid the effects associated with high-dimensional data analysis, this approach employs information on *hubness* to obtain data clustering solutions. This is the focus of the research work described herein. Thus, the fundamental concepts regarding the *hubness* aspect are presented in Section 2.1.

Still further, another aspect that can interfere in the expressivity of similarity measures arises when it becomes necessary to deal with more complex data domains, such as images. This is due to the existence of the so-called "*semantic gap*" between the low-level features that represent the data instances and the high-level of human perception, which can produce clusters obtained from information based only on low-level features that do not correspond to the notion of similarity of the user. Among such proposals for dealing with this question are the techniques for the semi-supervised clustering. More details concerning these techniques are described in Section 2.2.

2.1 The *hubness* aspect

Hubness is an aspect inherent to data that presents high dimensionality. As the intrinsic dimensionality of the data increases, the distribution of the *K-occurrences* in the list of *nearest neighbors* of each data instance becomes distorted and more variable. Hence, some data instances (called *hubs*) appear frequently in the list of *K-nearest neighbors*, and at the same time, other data instances (called *anti-hubs*) become infrequent neighbors. According to [Tomasev et al. 2014], the *hubness score*, the *hubs* and the *anti-hubs* are defined in the following manner:

—***Hubness score***: Let $X = \{x_1; ...; x_n\}$ be a set of data instances, $h_K(x)$ represents the number of $K-$*occurrences* of instances $x \in X$, that is, the number of times that x occurs in the list of K-*nearest neighbors* of other data instances which belong to X. Values commonly used for K are in the interval [5, 20] [Tomasev et al. 2011];

—$K-$***occurrences*** $h_K(x)$: Let $Y = \{y_1, y_2, ..., y_n\}$, be a set that contains n listings of K-*nearest neighbors* of data instances that belong to X, the number of K-*occurrences* $h_K(x)$ is the quantity of times that $x \in X$ occurs in listings of Y;

—***hubs***: Correspond to the data instances $x \in X$ that appear in many more listings of K-*nearest neighbors* than do the remaining data instances, that is, they possess *hubness scores* $h_K(x)$ significantly above the average;

—***anti-hubs***: Correspond to the data instances $x \in X$ that appear in very few listings of K-*nearest neighbors* of the remaining data instances, that is, they possess very low *hubness scores* $h_K(x)$ or even $h_K(x) = 0$.

According to [Tomasev et al. 2011], [Tomasev and Mladenic 2013] and [Tomasev et al. 2014], the *hubness* aspect can perform an important role in the task of clustering high-dimensional data, as it is a good reference to the centroids of the clusters. The main *hubs* can be used effectively as representatives to guide the process of assigning data instances to clusters, in a similar manner as occurs in techniques based on centroids.

Among the proposals for the use of the *hubness* aspect in the task of high-dimensional data clustering, one of the most promising is the *HPKM* (*Hubness-proportional k-means*) algorithm proposed in [Tomasev et al. 2011]. The *HPKM* is a variation of the traditional *k-means* algorithm, which employs a stochastic hybrid clustering approach, that is, it uses both *hubs* and centroids as representatives of clusters. The general idea of the algorithm consists of using the information concerning *hubness*

scores from the data instances to guide partitions in the first iterations and choosing a clustering configuration based on centroids at the end. As the conventional *k-means* algorithm, *HPKM* employs a single representative for each data partition.

Besides the good effectiveness demonstrated for dealing with high-dimensional data, the computational efficiency of the clustering algorithms based on *hubs* can be a cause of concern when one wishes to analyze large data sets. Through, considering the fact that the computational complexity of these algorithms is directly related to the *hubness score* calculation of the data instances, it is important to consider strategies that allow for the optimization of these calculations. Among such strategies, one can mention the use of an approximate *K-NN* graph for obtaining the *hubness score* with a time complexity of $\Theta(ndt)$, where n is the number of instances, d the quantity of dimensions and t represents the construction of the graph [Chen et al. 2009]. Another alternative pointed out in [Tomasev et al. 2014] consists of using approaches based on *locality-sensitive hashing* [Satuluri and Parthasarathy 2012].

It is also important to highlight that, as the *hubs* are centers of influence, possible inaccuracies associated with them can be easily propagated causing a negative impact on clustering. One way of dealing with this issue is to allow the incorporation of semi-supervised techniques into the data clustering process.

2.2 Semi-supervised data clustering

Different to the traditional approach for data clustering, the semi-supervised clustering approach counts on some additional information that guides data partitioning into clusters. This additional information can be obtained by means of labelling a small portion of the dataset [Chapelle et al. 2010] or informed in the form of data constraints [Davidson and Basu 2007][Xiong et al. 2014].

These constraints can be defined in different ways, for example: in instance level [Wagstaff and Cardie 2000], in feature level [Schmidt et al. 2011], in group level [Dubey et al. 2010], relative [Kumar and Kummamuru 2008] and ranked [Ahmed et al. 2012]. Among them, the instance level is the type that are most employed by clustering algorithms based on partitioning, thus significantly improving the performance of these algorithms [Dubey et al. 2010].

The instance level constraints are divided into two types, *must-link* and *cannot-link* constraints. A *must-link* constraint defines that a pair of instances must belong to the same cluster. On the other hand, a *cannot-link* constraint defines that a pair of instances must not belong to the same cluster [Wagstaff and Cardie 2000]. Formally, let us consider a set of data instances $X = \{x_1; ...; x_n\}$, a must-link constraint $r_{ml}(x_1, x_2)$ indicates that the instances x_1 and x_2 must be in the same cluster, whereas a cannot-link constraint $r_{cl}(x_1, x_3)$ indicates that the instances x_1 and x_3 must not be in the same cluster. It is important to highlight that the instance level constraints, although apparently simple, possess interesting properties. The *must-link* constraints, for example, are symmetric, reflexive and transitive. Another important question is related to the selection of data instances for obtaining the *must-link* and *cannot-link* constraints. When this selection is random, the resultant clusters may not reach the desired structure.

In the interest of resolving this limitation, *active learning* strategies are a viable option. The main goal of these strategies is to allow the selection of the most significant instances (or instance pairs) for labelling. This allows obtaining better clustering results while analyzing a small number of data instances [Li et al. 2014]. An example of a recent method that exploits the use of an active learning strategy in order to propose a semi-supervised learning approach can be seen in [Saito et al. 2014]. This method is based on the Optimum-Path Forest (OPF) methodology, where cluster assignment is performed computing an optimum connectivity with respect to a set of prototypes rather than computing local distance calculations [Amorim et al. 2016]. The active-learning strategy adopted in

[Saito et al. 2014] aims at improving the selection of informative labelled data instances and reducing the propagated errors on the unlabelled data instances considering a data classification problem.

The available approaches for the incorporation of constraints on the data clustering algorithms can be divided into two categories: those based on similarity and those based on search. The strategies that employ the approach based on similarity alter the similarity measure used in the clustering process, looking to satisfy the labels or constraints in the data in a manner that instances that should be together in the same cluster are close and instances that should not be together are separated. The strategies that employ the approach based on search use the labels or constraints supplied by the users in order to guide the clustering detection process for a data partition closer to the real dataset structure [Barioni et al. 2014]. The *SSHub Clustering* algorithm fits into the latter category.

3. *SSHUB CLUSTERING*

The clustering approach employed by the algorithm described herein combines semi-supervised and density estimation strategies based on *hubness score*, which aims at contributing toward the analysis of high-dimensional data. The main steps of the algorithm are presented in Section 3.1 and the constraint definition steps are discussed in Section 3.2.

3.1 Main steps

Given a dataset $X = \{x_1; ...; x_n\}$, a set H of *hubness scores* $h_K(x)$ for each instance x_i, and a percentage f of borderline elements that are to be analyzed for the definition of the *must-link* and *cannot-link* constraints. The *SSHub Clustering* (see Algorithm 1) employs a semi-supervised clustering approach based on the partitioning of X into a quantity of k clusters. Each cluster is represented by multiple prototypes, one main $p_i \in P$ and various auxiliaries $a_i \in A$. This representation strategy was adopted with the intention of allowing the algorithm to deal with clusters with hyperspherical shapes as well as with clusters that present arbitrary shapes.

In order to create the initial data partition, the *SSHub Clustering* algorithm starts out by selecting, in a semi-supervised manner, a main representative for each cluster (line 2 of the Algorithm 1). The strategy proposed for this initial phase of the algorithm allows guiding the user to select k main representatives from a ranking of the data instances with the highest *hubness score*. The remaining auxiliary representatives are derived from the *must-link* constraints $r_m(x_i, x_j) \in R_m$, starting from the second iteration of the algorithm. It is important to highlight that the initial main representatives are considered permanent cluster representatives, that is, even if they are updated in the following iterations, they remain as auxiliary representatives (line 5 of the Algorithm 1).

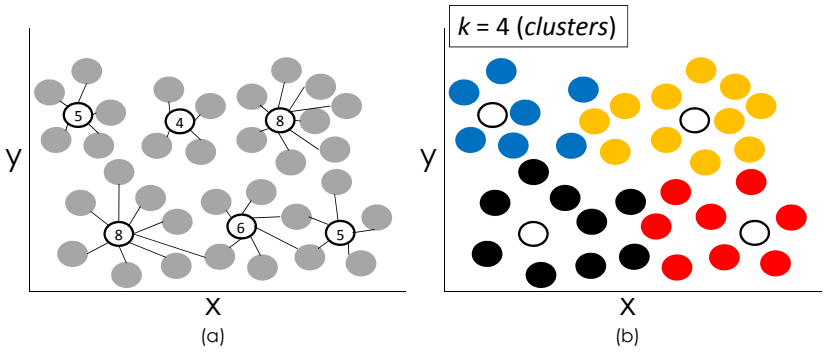

Fig. 2. Selection of $k = 4$ initial representatives. (a) *Hubs* (b) k selected *hubs* and the initial data partitioning.

Algorithm 1: *SSHub Clustering*

Data: Dataset X, Number of clusters k, *Hubness* H, Borderline instances f, MaxIterations $maxIt$
Result: C $\{c_1, ..., c_k\}$

```
1  begin
2      P ← Selection of k initial representatives;
3      A ← {}; R ← {}; N ← H; it ← 0;
4      for each representative p_i ∈ P do
5          A_i ← A_i ∪ p_i;
6      repeat
7          it++;
8          for each instance x_i ∈ X do
9              Assign x_i to the cluster with the smaller aggregated distance regarding both main and
               auxiliary prototypes; if x_i do not change cluster then
10                 N_i ← N_i^2;
11             else
12                 N_i ← H_i;
13         for each c_i do
14             Update the representative p_i;
15         ConstraintDefinition(C, H, f, A, R);
16     until convergence or it = maxIt ;
17     return C
18 end
```

Figure 2 illustrates the initial selection phase of the *SSHub Clustering* algorithm considering $K = 5$ *nearest neighbors* for calculating the *hubness score* ($h_K(x)$) of each data instance. Considering the existence of various *hubs* in the dataset (white circles with the h_K values in Figure 2(a)), the idea of the strategy adopted in this phase is to provide the possibility to the user of interfering at the outset of the clustering process for selecting the initial k representatives from among these *hubs* (Figure 2(b)). This algorithm feature is especially useful in the analysis of complex data domains that can be affected by the "*semantic gap*" problem. Figure 3 presents a practical example of the selection of initial representatives in image data domains. Considering the existence of an application, which exhibits thumbnails that correspond to the higher *hubness score* instances (calculated based on low level features, such as color, texture or shape) from an image dataset, a user can, based on their data domain knowledge, select a initial representative for each desired cluster.

After the selection of the initial representatives, the assignment of the instances to the clusters is performed (line 9 of Algorithm 1) by taking into account main and auxiliary-representatives. Both types of representatives are obtained through *hubness* information in order to effectively deal with high-dimensional data. In this phase, the *SSHub Clustering* algorithm considers the aggregated distance calculated as defined in Equation 1, while respecting the previously informed *must-link* and *cannot-link* constraints. In this equation, Q_k is the set of representatives (main and auxiliary) of a cluster c_k, and $\delta()$ is a distance function. It is important to emphasize that in the first iteration of the algorithm, there do not exist constraints and auxiliary representatives. This information is considered only from the second iteration of the algorithm.

$$\delta_g(Q_k, x_i) = \min_{q_j \in Q_k} (\delta(q_j, x_i)) \tag{1}$$

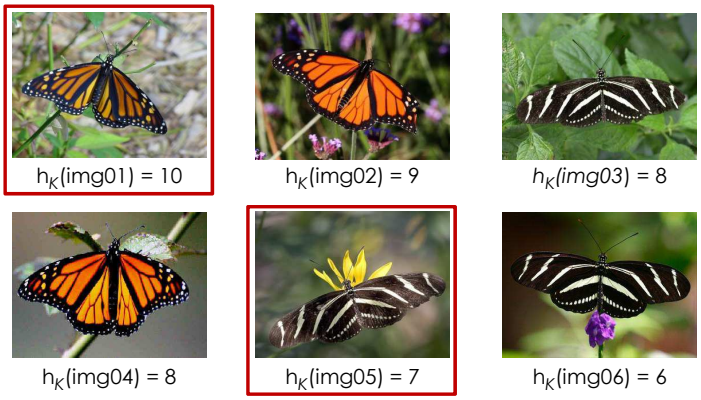

Fig. 3. Selection of k ($k = 2$) initial representatives in an image dataset. Adapted from [Wang et al. 2009].

The updating of the main representatives of each cluster is based on the *hubness score* of the instances that belong to each cluster (lines 10 to 15 of Algorithm 1). However, this score can be changed over the iterations of the algorithm. Inspired upon the approach put forward for the algorithm *HPKM* [Tomasev et al. 2014], instances that do not change cluster over the iterations are privileged through the squared elevation of their *hubness score* (line 11 of Algorithm 1). Should there be noted changes in the cluster, the *hubness score* returns to its original value (line 13 of Algorithm 1). In this manner, the representative of each cluster corresponds to the instance with the highest accumulated *hubness score* (line 15 of Algorithm 1).

3.2 Constraint definition steps

The constraint definition phase of the *SSHub Clustering* (line 16 of Algorithm 1) aims at selecting a subset of instances in such a way that to find additional knowledge about them allows obtaining a more adequate data partitioning. Therefore, the f instances farthest from each $p_i \in P$ with a *hubness score* ≥ 1 are evaluated. This restriction to the *hubness score* is to avoid the possibility of selecting *anti-hubs*. The sub-routine responsible for this task is described in Algorithm 2.

The definition of the quantity of instances q that should be analyzed in each cluster, in the constraint definition phase, is performed in a manner that is proportional to the quantity of instances assigned to each cluster (line 4 of Algorithm 2). Considering each borderline instance x_y of a given cluster c_i, the constraint definition sub-routine selects the nearest instance $x_w \in c_j$ to x_y, such that $c_i \neq c_j$. Through such, it is possible to analyze distinct pairs of borderline instances in order to specify whether or not the elements must belong to the same cluster. This additional information can be obtained from labels or cluster-guiding constraints fed by users. This analysis strategy allows to create *must-link* constraints $r_{ml}(x_y, x_w)$ (line 14 of Algorithm 2) or *cannot-link* constraints $r_{cl}(x_y, x_w)$ (line 16 of Algorithm 2).

Moreover, to obtain auxiliary representatives for the clusters, each x_y of a given cluster c_i is analyzed to specify if x_y and p_i must be in the same cluster or not. If the answer is positive, a *must-link* constraint $r_{ml}(x_y, p_i)$ is created (line 9 of Algorithm 2) and x_y is attributed to A (line 10 of Algorithm 2). On the other hand, a *cannot-link* constraint $r_{cl}(x_y, p_i)$ is created (line 12 of Algorithm 2). Finally, constraints are created via transitivity (line 17 of Algorithm 2). For example, if there exist the $r_{ml}(p_i, x_y)$ and the $r_{ml}(x_y, x_w)$ constraints, then the $r_{ml}(p_i, x_w)$ constraint should also exist.

Figure 4 illustrates the four possible scenarios of the analysis process for the borderline elements (see Algorithm 2). Figure 4(a) presents the scenario in which only the *must-link* constraints are generated, that is, p_i, x_y and x_w should be in the same cluster. In this scenario, the $r_{ml}(x_y, x_w)$ constraint

Algorithm 2: ConstraintDefinition

Data: Clusters C, *Hubness* H, Borderline instances f, Representatives A, Constraints R

begin
  $t \leftarrow |N| \cdot f$;
  for *each* $c_i \in C$ **do**
    $q \leftarrow (|c_i|/|N|) \cdot t$;
    for $j \leftarrow$ *1 until* $j = q$ **do**
      $x_j \leftarrow$ select the farthest instance from the main prototype p_i in c_i, with $h_K \geq 1$;
      $x_w \leftarrow$ select the nearest instance from x_j in $c_w \mid w \neq i$;
      if p_i *and* x_j *must be in the same cluster* **then**
        $R \leftarrow R \cup r_{ml}(p_i, x_j)$;
        $A_i \leftarrow A_i \cup x_j$;
      else
        $R \leftarrow R \cup r_{cl}(p_i, x_j)$;
      if x_w *and* x_j *must be in the same cluster* **then**
        $R \leftarrow R \cup r_{ml}(x_w, x_j)$;
      else
        $R \leftarrow R \cup r_{cl}(x_w, x_j)$;
      Define transitive constraints;
      j++;
end

indicates the need of a displacement of x_w. Figure 4(b) presents the scenario in which a *must-link* constraint is generated between the borderline instance x_y with its main respective representative p_i, and a *cannot-link* constraint between the borderline instance x_y and the instance from another cluster x_w, or be it, p_i and x_y should be in the same cluster and x_y and x_w should not be in the same cluster.

Figure 4(c) presents the scenario in which a *cannot-link* constraint is generated between the borderline instance x_y and its main respective representative p_i and a *must-link* constraint is generated between the borderline x_y and the instance x_w from another cluster. In this case, the two borderline instances x_y and x_w should be in the same cluster ($r_{ml}(x_y, x_w)$) indicating the need of a displacement of x_y. Figure 4(d) presents the last scenario in which only *cannot-link* constraints are generated. In this scenario, a *cannot-link* constraint is generated between the borderline instance x_y and its main respective representative p_i and a *cannot-link* constraint between the borderline instance x_y and the instance x_w of another cluster, or be it, p_i, x_y and x_w should not be part of the same cluster.

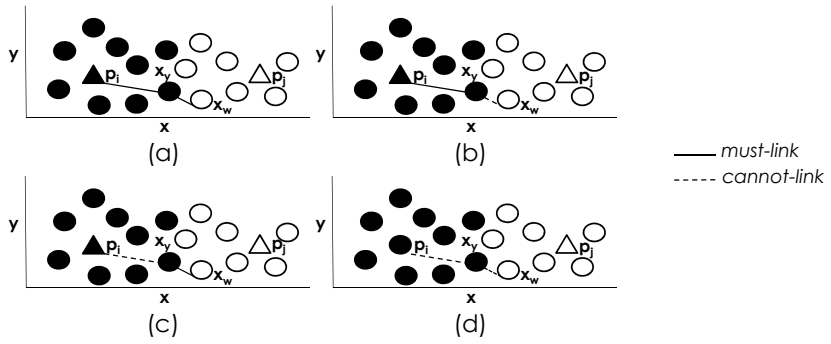

Fig. 4. Possible scenarios for creating *must-link* and *cannot-link* constraints.

In summary, the steps performed by the *SSHub Clustering* algorithm are presented below:

(1) Select k instances with the higher *hubness scores* as the initial representatives $P = \{p_i, ..., p_k\}$ and make $A = P$;

(2) Assign each instance $x_i \in X$ to the cluster with the smaller aggregated distance δ_g in relation to Q, respecting the constraints R_m and R_c;

(3) Update the main representatives P of each cluster based on the *hubness scores* of the instances assigned to each cluster;

(4) Analyze the f borderline instances to define *must-link* and *cannot-link* constraints and other auxiliary representatives (Algorithm 2);

(5) Return to step (2) until convergence or until reaching the maximum number of iterations.

The stop criterion for the algorithm *SSHub Clustering* (step 5) checks two possibilities: 1) if the maximum number of iterations was reached or 2) if the convergence of the algorithm was obtained. The convergence function considered verifies the sum of the distances for the representatives (main and auxiliary) for the current iteration in relation to the previous iteration. Equation 2 presents how this calculation is performed:

$$\sum_{k=1}^{k} \sum_{x_i \in C_k} \min_{q_j \in Q_k} \left(\delta(q_j, x_i)\right), \tag{2}$$

where x_i is an instance of a cluster C_k, Q_k is a set of representatives (main and auxiliary) of the respective cluster, and $\delta()$ is a distance function.

Assuming that the *hubness scores* are already computed, the time complexity of the constraint definition phase (Algorithm 2) at each iteration of *SSHub Clustering* is $O(|C| \cdot |R|^2)$, where $|C|$ is the cardinality of the dataset and $|R|$ is the cardinality of the constraint set. The time complexity of the remaining steps of the algorithm is $O(it \cdot |Q| \cdot |X| \cdot d)$, where it is the number of iterations, $|Q|$ is the cardinality of the set of representatives (main and auxiliaries), $|X|$ is the number of data elements and d is the number of attributes.

4. EXPERIMENTS

We employed 23 real datasets to perform the experiments. The details concerning each one are presented in Table II. From these datasets, 22 were obtained from *UCI Machine Learning* [Lichman 2013] and the *NBA* dataset was obtained from the repository BasketballStats [Databasesports.com 2011]. It is important to highlight that the datasets were selected based on their different number of classes, densities and dimensions, with the aim of checking the performance of the proposed algorithm against other four algorithms described in the literature in different scenarios.

All these datasets were preprocessed according to the recommendations indicated in these repositories. In addition, for the dataset Covertype (5) a sample with 19,229 randomly selected instances from the original set was considered, maintaining the proportion of instances per class. For the dataset Hepmass (8) a sample of 45,000 instances from the original set was considered, equally divided between the two classes. Finally, for the dataset Digits389 (6) only the instances of classes 3, 8 and 9 from the dataset Pendigits were considered, as was performed in [Xiong et al. 2014].

The experiments were run on a computer equipped with a processor of the following specifications: Intel Core i5-3230M CPU @ 2.60GHz with 6 GB of RAM, hard drive 500 GB SATA (7,200 rpm)

Table II. Datasets employed in the experiments.

ID	Datasets	# of data instances	# of dimensions	# of classes
1	Abalone	4,176	8	3
2	Aloi	1,098	64	10
3	Breast	683	9	2
4	Churn	5,000	14	2
5	Covertype	19,229	54	7
6	Digits389	3,165	16	3
7	Ecoli	336	7	8
8	Hepmass	45,000	27	2
9	Isolet	6,237	617	26
10	Musk2	6,598	166	2
11	NBA	22,064	17	2
12	Page blocks	5,473	10	5
13	Pendigits	10,992	16	10
14	Segment	2,310	19	7
15	Sonar	208	60	2
16	Spambase	4,599	57	2
17	Urbanlandcover	675	147	9
18	Vehicle	846	18	4
19	Vowel	990	13	11
20	Wine	178	13	3
21	WDBC	569	30	2
22	Yeast	1,484	8	10
23	Zoo	101	16	7

running Microsoft Windows 7 64 bits. The algorithms *HPKM*, *Kernel k-means* [Dhillon et al. 2004], *DBSCAN* [Sander et al. 1998] and *SSDBSCAN* [Li et al. 2014] were selected as the baselines for comparison. The *hub* based *HPKM* was the clustering algorithm that inspired the proposal of the *SSHub Clustering* algorithm. The *Kernel k-means* is a classic representative of kernel based clustering algorithms, which are known for dealing well with non-hyperspherical clusters, and the *DBSCAN* is a standard representative of clustering algorithms based on density. As a representative of the semi-supervised clustering algorithms the extension of the *DBSCAN*, called *SSDBSCAN* [Li et al. 2014], was chosen. This algorithm incorporates additional information in the form of labels in the clustering process based on density. The implementation of this algorithm was made available by the authors of the work-study and was written in the Java language. The implementation of all the other algorithms (*HPKM*, *Kernel k-means* and *DBSCAN*) were made available by the authors of the work-study described in [Tomasev et al. 2014] in repository [Tomasev 2014], all written in Java. With the aim of allowing for a fair comparison of the algorithms, *SSHub Clustering* was also implemented using Java.

Considering that the ideal neighborhood size that should be analyzed in the calculation of the *hubness score* may vary according to the data domain, four distinct neighborhood values were considered (K = 5, 10, 15 and 20) for the *SSHub Clustering* and *HPKM* algorithms. The number of requested clusters (k) was defined according to the number of classes defined for each dataset and described in Table II. All the algorithms take into consideration the Euclidean distance function.

In order to automate the execution of the experiments with the *SSHub Clustering* algorithm, the user interaction phase for choosing the initial representatives was simulated in the following manner. Considering the label information of the datasets, the initial representatives were chosen among the five instances of each cluster with the highest *hubness scores*. In this manner, it is possible to simulate the fact that at each run, the user can choose different instances of high *hubness* scoring as representatives. In addition, the total quantity of borderline elements (f) analyzed by the *SSHub Clustering* was defined at 1% of the total instances. The same value of 1% was considered for the instances labeled from the *SSDBSCAN* algorithm. This limit for semi-supervised information was defined at 1% due to the fact that in real applications, where the user interacts with the semi-supervised clustering process, it is not expected that the user be required to provide a high quantity of constraints.

As shown in [Tomasev et al. 2011], the use of *hubs* as initial representatives results in a rapid convergence of the algorithm, i.e. only a few iterations are necessary for the algorithm to converge. This fact was also observed in trials conducted on the work-study described herein. Therefore, the maximum number of iterations for the *SSHub Clustering* algorithm was defined as 15. The values for the other parameters required by the *DBSCAN* algorithm, EPS and MinPts, were selected with an automatic routine tailored to specify appropriate parameter values for each data set. This routine was made available in [Tomasev 2014]. For the *SSDBSCAN* algorithm there is no need of specifying the EPS value. The selection of values for the MinPts parameter varied from 3 up to 30, depending on the size of the dataset.

The main objective of the experiments was to evaluate and to compare the effectiveness of the algorithm in different scenarios, especially in those where there exist a higher number of dimensions. In this context, it was necessary to use an evaluation measure that allowed for measuring to which extent an algorithm is capable of finding the true structure of a dataset. By considering the existence of knowledge regarding the desired partitioning for the datasets, the Adjusted Rand Index was used [Hubert and Arabie 1985] for evaluating the effectiveness of the algorithms. In this index, the quality of the results is represented in the interval $[-1, 1]$, being that the closer it is to 1, the better will be the correspondence between the obtained cluster and the desired partitioning for the evaluated dataset.

As the algorithms employed in the experiments present nondeterministic startup features, they were run 50 times for each evaluation performed. Therefore, each presented measurement considers an average of 50 algorithm runs, without considering the best or the worst for each of them. Sections 4.1 and 4.2 present the analyzes of the results obtained throughout the experiments.

4.1 Comparison of the algorithms

This section presents the results of the experiments that allow for the comparison of the *SSHub Clustering* algorithm effectiveness with regard to *HPKM*, *Kernel k-means*, *DBSCAN* and *SSDBSCAN* algorithms. Moreover, the time required by each one of the algorithms during the running of these experiments are also presented. The results achieved are shown in Table III and Figures 5 and 6. In this table, the labels *KerKM* and *SSDBS* represent the *Kernel k-means* and the *SSDBSCAN* algorithms, respectively.

Analyzing the results shown in Table III, one notes that the semi-supervised algorithms *SSHub Clustering* and *SSDBSCAN* present higher effectiveness in 17 of the 23 tested datasets (and two more draws in the first place). And of these 17 datasets, the *SSHub Clustering* presented a superior effectiveness in 9 of them. These results help to corroborate the claim that semi-supervised approaches help to obtain a data partitioning closer to the real dataset structure.

It is also possible to notice that the neighborhood defined for the *hubness score* calculation can impact on the partitions generated by the *SSHub Clustering*. This fact can be seen mainly by observing sets (2), (6), (7), (8), (11), (13), (14), (17), (20), (21) and (23). The same behavior is not evident in the results presented by the *HPKM* algorithm. Therefore, it becomes necessary to investigate to which extent each one of the strategies employed by the *SSHub Clustering* collaborate toward obtaining good results. The results of this evaluation are presented in Section 4.2.

Aiming to verify if there exists a significant difference between the effectiveness of the evaluated algorithms, the *Friedman* [Demsar 2006] and the *post-hoc of Bonferroni-Dunn* [Zar 2007] statistics tests were used. Both tests were applied in order to discover if with a 95% of certainty one can conclude that the *SSHub Clustering* surpasses the *HPKM*, *Kernel k-means*, *DBSCAN* and *SSDBSCAN*.

The *Friedman* and the *Bonferroni-Dunn* tests are based on the comparison of performance ranks (see Table III). To apply the *Friedman* test, we consider the sum of the *ranks* (R) of each algorithm for the calculation of F_F such that:$F_F = ((N-1)\chi_F^2)/(N(A-1)-\chi_F^2)$, where $\chi_F^2 = (12/(NA(A+$

Table III. Experimental results. The values in bold highlight the best performances. The numbers between ( ) show the position in the ranking.

	SSHub Clustering				***HPKM***						
Dataset	$K=5$	$K=10$	$K=15$	$K=20$	$K=5$	$K=10$	$K=15$	$K=20$	***KerKM***	***DBSCAN***	***SSDBS***
1	0.08	**0.17 (1)**	0.10	0.11	0.14 (2.5)	0.14	0.14	0.13	0.00 (4.5)	0.00 (4.5)	0.14 (2.5)
2	0.65	0.61	0.62	**0.72 (1)**	0.53	0.52	0.53	0.54 (3)	0.00 (5)	0.28 (4)	0.68 (2)
3	**0.90 (1)**	0.88	0.86	0.85	0.85 (2)	0.85	0.85	0.85	0.78 (4)	0.01 (5)	0.84 (3)
4	-0.02	**0.15 (1)**	0.03	0.02	-0.09	-0.09	-0.09	-0.05 (5)	0.00 (3.5)	0.00 (3.5)	0.10 (2)
5	0.20	0.23	0.24 (2)	0.19	0.18	0.19	0.19	0.20 (3)	0.00 (5)	0.11 (4)	**0.37 (1)**
6	0.10	0.74	0.79 (2)	0.47	0.33	0.54 (4)	0.33	0.35	0.00 (5)	0.66 (3)	**0.95 (1)**
7	0.70	0.73	0.63	**0.74 (1)**	0.44	0.42	0.43	0.49 (2)	0.00 (5)	0.06 (4)	0.31 (3)
8	0.32 (2)	0.22	0.30	0.19	0.33	0.32	**0.34 (1)**	0.32	0.00 (4.5)	0.00 (4.5)	0.05 (3)
9	**0.52 (1)**	0.49	0.49	0.50	0.43	0.40	0.44 (2)	0.43	0.00 (5)	0.02 (4)	0.39 (3)
10	0.07 (2)	-0.01	-0.01	-0.02	-0.06	-0.03 (5)	-0.04	-0.03	0.02 (3)	0.00 (4)	**0.27 (1)**
11	-0.01	**0.74 (1)**	-0.03	0.63	0.00 (5)	-0.01	0.00	0.00	0.07 (4)	0.13 (3)	0.14 (2)
12	-0.04	0.11	0.10	0.17 (3)	0.10	0.11 (4)	0.08	0.10	0.00 (5)	0.36 (2)	**0.55 (1)**
13	0.67 (2)	0.49	0.56	0.54	0.54	0.55	0.56 (3)	0.53	0.00 (5)	0.37 (4)	**0.87 (1)**
14	0.37	0.50 (3)	0.46	0.50	0.48	0.48	0.51 (2)	0.48	0.00 (5)	0.28 (4)	**0.65 (1)**
15	0.02	0.00	0.02	**0.03 (1)**	0.00	0.00	0.00	0.01 (2)	0.00 (4)	0.00 (4)	0.00 (4)
16	-0.03	-0.01	-0.01	0.00 (4)	-0.01 (5)	-0.01	-0.03	-0.02	0.10 (2)	0.03 (3)	**0.22 (1)**
17	0.08	0.42 (2)	0.42	0.42	0.44	0.44	**0.51 (1)**	0.42	0.00 (5)	0.06 (4)	0.23 (3)
18	0.13 (2)	0.07	0.08	0.09	0.08 (3)	0.08	0.08	0.07	0.00 (4.5)	0.00 (4.5)	**0.17 (1)**
19	0.05	0.07	**0.09 (1.5)**	0.06	0.07	0.07	0.08 (3)	0.08	0.00 (5)	**0.09 (1.5)**	0.04 (4)
20	0.37	0.83	0.86 (2)	0.77	**0.91 (1)**	0.85	0.86	0.85	0.00 (4)	-0.01 (5)	0.45 (3)
21	0.60	0.73 (2)	0.19	0.26	**0.74 (1)**	0.74	0.74	0.71	0.00 (5)	0.01 (4)	0.30 (3)
22	**0.16 (1.5)**	0.15	0.15	0.15	0.14	0.15	**0.16 (1.5)**	0.15	0.00 (5)	0.02 (4)	0.08 (3)
23	0.80	**0.96 (1)**	0.91	0.91	0.68	0.61	0.67	0.77 (2)	0.73 (3)	0.40 (4)	0.10 (5)
Sum *ranks*	40				63				101	87.5	53.5
AVG *ranks*	1.74				2.74				4.39	3.8	2.32

1)). $\sum_{i=1}^{A} R_i^2 - (3N(A+1)))$, N is the number of datasets and A is the number of tested algorithms. As the F_F statistic (19.46) is greater than the critical value *FDistribution* (2.47), this result shows that there exists a significant difference among the effectiveness of the algorithms.

In pursuance of indicating which of the algorithms had the best performance, the *post-hoc Bonferroni-Dunn* test was used, setting *SSHub Clustering* as the control algorithm. In this test, the performance of two algorithms is statistically different, if the difference between the average *ranks* divided by $\sqrt{\frac{A(A+1)}{6N}}$ is greater or equal to the critical difference (CD), which is calculated as: $CD = q_\alpha \sqrt{\frac{A(A+1)}{6N}}$, where the critical value q_α (2.498) is obtained from the table found in [Demsar 2006]. With the CD value calculated (1.16), one observes that the differences between the *rank* averages of the algorithms *SSHub Clustering* and *HPKM* (2.14), *SSHub Clustering* and *Kernel k-means* (5.68), *SSHub Clustering* and *DBSCAN* (4.42) and *SSHub Clustering* e *SSDBSCAN* (1.25) were higher than CD. Therefore, it is possible to conclude that the effectiveness of the *SSHub Clustering* algorithm is higher than that of the algorithms selected as the baseline for the tested datasets.

In addition to the effectiveness, the efficiency of the algorithms was also analyzed. Figures 5 and 6 presents the times (in milliseconds) spent by each algorithm for obtaining the results presented in Table III. Figure 5 shows the graphs of the datasets (1) to (12) and Figure 6 shows the graphs of the datasets (13) to (23). The following labels were used in these figures to represent the results obtained for each algorithm on the x-axis: *SSHub Clustering* with $K = 5$ (1), *SSHub Clustering* with $K = 10$ (2), *SSHub Clustering* with $K = 15$ (3), *SSHub Clustering* with $K = 20$ (4), *HPKM* with $K = 5$ (5), *HPKM* with $K = 10$ (6), *HPKM* with $K = 15$ (7), *HPKM* with $K = 20$ (8), *Kernel k-means* (9), *DBSCAN* (10) and *SSDBSCAN* (11).

The time spent by the *HPKM* and the *SSHub Clustering* algorithms considers the time required for calculating the *hubness score* of the datasets (see the striped bars) and the execution time of the algorithms. The time for calculating the *hubness score* was computed for the different neighborhoods under consideration, i.e. (5), (10), (15) and (20). This time was considered separately, since it is a pre-processing stage of the *HPKM* and *SSHub Clustering* algorithms.

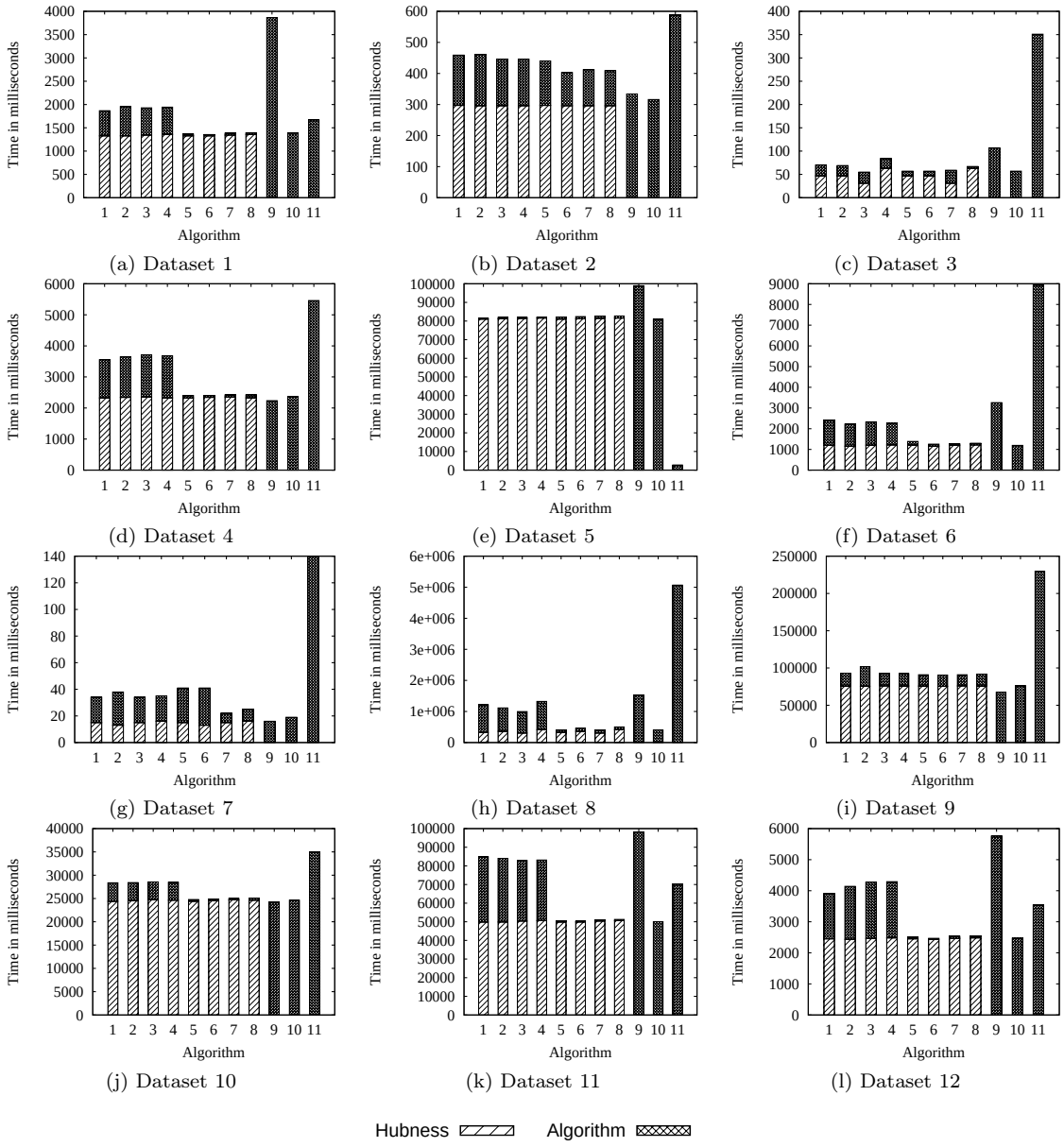

Fig. 5. Execution time (milliseconds) of each algorithm for the 1-12 datasets. Bars 1-4: *SSHub Clustering*; bars 5-8: *HPKM*; bar 9: *Kernel k-means*; bar 10: *DBSCAN*; bar 11: *SSDBSCAN*. The striped bars represent the time to compute the *hubness* score of the datasets.

It is possible to note that the *HPKM* algorithm presents the lowest run times in 19 of the 23 tested datasets. It was expected that through the comparison of the *HPKM* and the *SSHub Clustering* algorithms, the *HPKM* would present a lower run time, since it does not possess the overhead of the semi-supervised approach. However, it is important to highlight that in spite of this fact, *SSHub Clustering* demanded less time than the *HPKM* in 4 of the evaluated datasets (sets 5, 15, 20 and 23). Moreover, when we consider the trade-off between efficiency and effectiveness it is important to note that *SSHub Clustering* presented a better effectiveness than the *HPKM* in 17 of the evaluated datasets (see Table III).

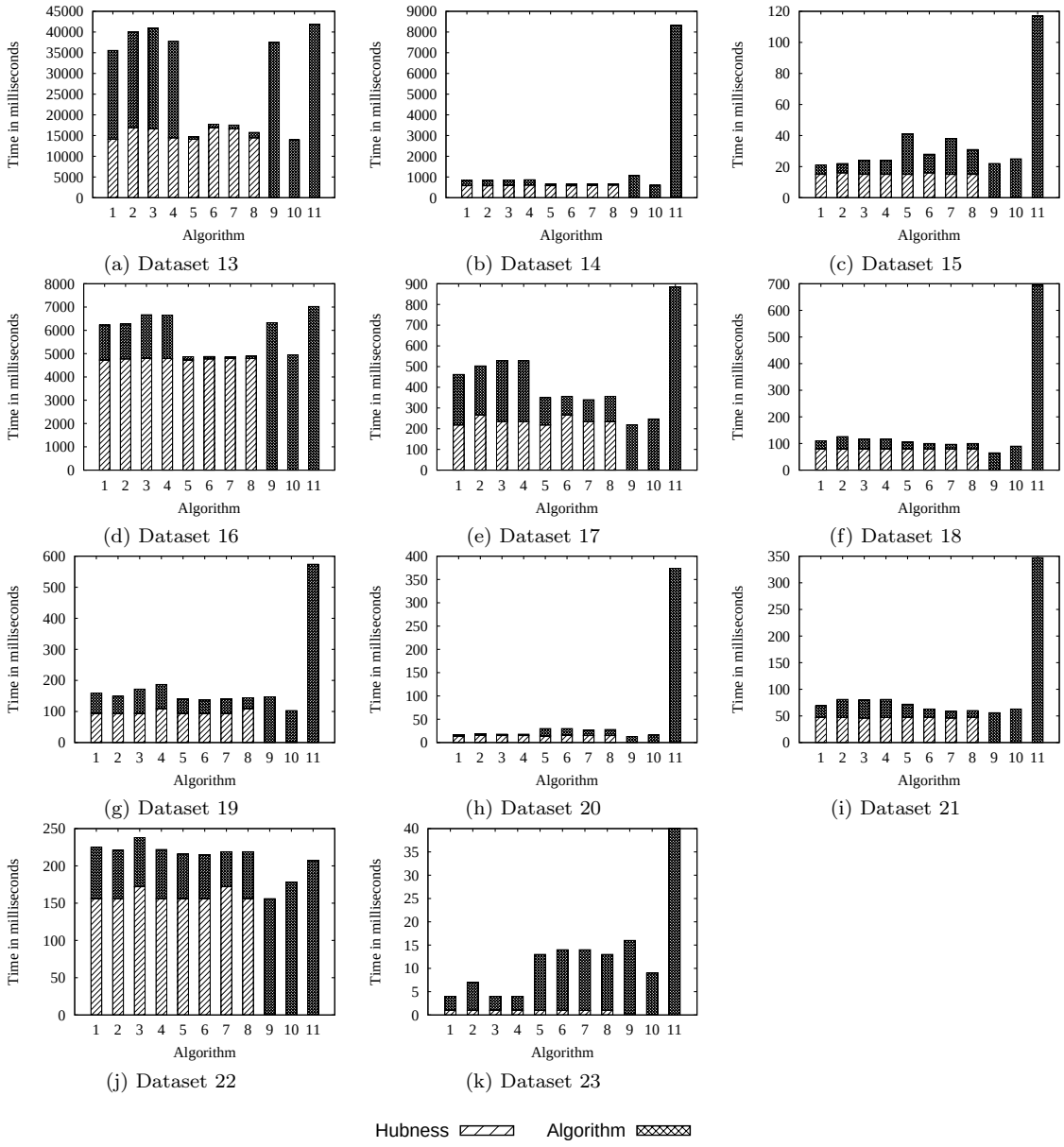

Fig. 6. Execution time (milliseconds) of each algorithm for the 13-23 datasets. Bars 1-4: *SSHub Clustering*; bars 5-8: *HPKM*; bar 9: *Kernel k-means*; bar 10: *DBSCAN*; bar 11: *SSDBSCAN*. The striped bars represent the time to compute the *hubness* score of the datasets.

Analyzing the performance of the *SSHub Clustering* algorithm regarding the others (that is, not considering the *HPKM* algorithm), it is possible to verify that it presents the lowest run times in 18 of the 23 evaluated datasets. Moreover, when only the semi-supervised clustering approaches *SSHub Clustering* and *SSDBSCAN* are considered, the *SSHub Clustering* algorithm presents the lowest run times for all datasets. Noteworthy here is that, even when considering the total sum of the time spent with the *hubness score* calculation to the processing time of the *SSHub Clustering* algorithm, it demanded less time than the *SSDBSCAN* algorithm in 19 of the 23 datasets. For example, for the dataset 8, the *SSHub Clustering* algorithm demanded 74% less time than the *SSDBSCAN* and obtained a better effectiveness (see Table III).

4.2 Varying the parameters

The experimental results presented in Section 4.1 show that the *SSHub Clustering* algorithm has a higher effectiveness than all the other algorithms considered in the evaluation and an efficiency higher than three of these algorithms. It loses only to the *HPKM* that employs an unsupervised clustering approach. In addition, these results also show a variation in the effectiveness of the *SSHub Clustering* algorithm when the neighborhood defined by the *hubness score* calculation varies. With the aim of investigating the individual contribution of each strategy employed by the *SSHub Clustering* algorithm in the final result of the clustering experiments described herein, two variations of the parameters were considered: the non-updating of the main representatives (Section 4.2.1) and the different percentages of borderline elements (Section 4.2.2).

To perform these experiments, 10 of the 23 datasets were selected, which were used in the experiments described in Section 4.1 (see Table III): Churn (4), Covertype (5), Digits389 (6), Hepmass (8), Isolet (9), Musk2 (10), NBA (11), Pendigits (13), Segment (14) and Urbanlandcover (17). These sets were selected while taking into consideration that, in general, they present the highest quantity of instances and greater variability of classes and dimensions.

4.2.1 *Non-updating of the main representatives .* The original version of the *SSHub Clustering* presented in Algorithm 1 (lines 10 to 15) considers the updating of the main representatives at each iteration. However, as this phase is unsupervised, it is possible for the main representative to be replaced by a data instance which does not have the same label as the main representative of the previous iteration.

Therefore, for these experiments, we wanted to verify if the updating of the main representatives could have a negative impact, in terms of the effectiveness of the algorithm, as it may diverge from the information obtained through the selection of the initial representatives during the interaction phase with the user. For this investigation, a new version of the algorithm was implemented that does not update the main representatives at each iteration. This version only maintained the update for the auxiliary representatives that are derived from the *must-link* constraints.

Table IV presents the results of this evaluation. Column (1) represents the original version of the *SSHub Clustering* algorithm, and column (2) the modified version that does not consider the updating of the main representatives.

Observing Table IV it is noted that for values lower than $K = 5$, the new version without the updating of the main representatives surpasses the original version in 7 out of the 10 evaluated datasets. Considering $K = 20$, the new version presents an effectiveness that is superior for every one of the 10 evaluated datasets. As the value of K increases, the number of datasets for which the new version presents superior results increases and the difference in the effectiveness also increases.

Table IV. Comparison of *SSHub Clustering* algorithm with (1) and without (2) updating the main representatives. Adjusted Rand Index values.

	***SSHub Clustering* (1)**				***SSHub Clustering* (2)**			
Dataset	$K = 5$	$K = 10$	$K = 15$	$K = 20$	$K = 5$	$K = 10$	$K = 15$	$K = 20$
4	-0.02	**0.15**	0.03	0.02	0.03	**0.15**	0.11	0.10
5	0.20	0.23	0.24	0.19	0.22	0.24	**0.25**	0.24
6	0.10	0.74	0.79	0.47	0.50	0.95	0.89	**0.97**
8	**0.32**	0.22	0.30	0.19	0.23	0.22	0.17	0.25
9	0.52	0.49	0.49	0.50	0.28	0.55	**0.56**	0.54
10	0.07	-0.01	-0.01	-0.02	0.09	0.20	0.12	**0.25**
11	-0.01	0.74	-0.03	0.63	0.14	0.60	0.61	**0.75**
13	0.67	0.49	0.56	0.54	0.55	**0.79**	0.73	0.76
14	0.37	0.50	0.46	0.50	0.52	0.59	0.59	**0.61**
17	0.08	0.42	0.42	0.42	0.43	0.48	0.47	**0.49**

Table V. Comparison of *SSHub Clustering* algorithm with different percentages for borderline instances. Adjusted Rand Index values.

	SSHub Clustering			
Dataset	0%	1%	5%	10%
4	0.04	0.15	0.20	**0.33**
5	0.15	0.23	0.20	**0.27**
6	0.59	0.74	0.92	**0.95**
8	0.14	0.22	0.20	**0.27**
9	0.46	0.49	0.54	**0.55**
10	-0.01	-0.01	-0.06	**0.04**
11	0.04	0.74	0.75	**0.79**
13	0.50	0.49	0.53	**0.62**
14	0.45	0.50	0.50	**0.66**
17	0.46	0.42	0.45	**0.49**

This behavior can be explained by the fact that when the algorithm considers a smaller neighborhood (for example, $K = 5$) for the identification of *hubs*, these may not initially be so representative. Therefore, in this case, the final clustering result can be improved through the approach that allows for the updating of main representatives at each iteration. On the other hand, when the algorithm considers a larger neighborhood (for example, $K = 20$), the *hubs* become more representative and the selection of the initial representatives at the interaction phase with the user becomes sufficient for obtaining good results.

4.2.2 *Percentage variation of borderline instances.* The clustering approach employed by the *SSHub Clustering* algorithm (see Section 3) allows for semi-supervision in two moments: during the selection of the initial representatives and during the analysis of the borderline elements. The experiments described in this section aim to allow the evaluation of the contribution made in this phase of the analysis of borderline elements to the final clustering result. Hence, Table V exhibits the results of the experiments by considering the parameter f as varying from 0 to 10% of the dataset size. Noteworthy here is that when $f = 0\%$ none of the borderline instances are analyzed, i.e., semi-supervision is only considered in the definition of the initial representatives at the first iteration of the algorithm. For the experiments shown herein, $K = 10$ was adopted for the calculation of the *hubness score*.

Through the analysis of the results presented in Table V, one notes that, in a general sense, the rate at which the value of the parameter f increases, the effectiveness of the algorithm also increases (i.e., the value of the Adjusted Rand Index increases). For example, for dataset 4, the increase in effectiveness (considering f varying from 0 to 10%) varied from 3.75 to 8.25 times when compared to the result obtained through running the *SSHub Clustering* algorithm with $f = 0$ and for dataset 11, the increase in effectiveness varied from 18.50 to 19.75 times when compared to the results obtained with $f = 0$.

5. CONCLUSIONS

The combination of *hubness* with semi-supervision strategies to allow clustering results, which match best with the desired partitioning for high-dimensional data, had still not been explored in the related works. The proposed *SSHub Clustering* algorithm employs a semi-supervised clustering approach based on partitioning, in which each partition is represented by multiple prototypes with its definition based on *hubness score* information of the data instances. This information is used by the *SSHub Clustering* semi-supervision strategies at two moments: during the selection of the initial representatives and during the analysis of borderline elements.

The experimental results demonstrate that the method developed herein statistically surpasses four approaches proposed in the literature that present good results in the treatment of high-dimensional data; those being *HPKM*, *Kernel k-means*, *DBSCAN* and *SSDBSCAN*. Moreover, it was shown that

the proposed algorithm possesses great potential for dealing with different quantities of instances, clusters and dimensions. Among the planned future work, the exploration of other strategies for the selection of borderline elements in the semi-supervision phase is prioritized, as well as work toward improving the computational efficiency of the method for calculating the *hubness scores*, taking as a base the work described in Section 2. Moreover, we intend to expand the scope of the experiments by comparing the proposed algorithm with other approaches, such as: graph-based clustering and semi-supervised algorithms based on the optimum-path forest.

REFERENCES

Aggarwal, C. C. and Reddy, C. K. *Data Clustering: Algorithms and Applications*. Chapman & Hall/CRC, 2013.

Aggarwal, C. C. and Yu, P. S. Finding generalized projected clusters in high dimensional spaces. In *ACM SIGMOD International Conference on Management of Data*. Dallas, Texas, pp. 70–81, 2000.

Ahmed, E. B., Nabli, A., and Gargouri, F. Shacun: semi-supervised hierarchical active clustering based on ranking constraints. In *Industrial Conference on Advances in Data Mining*. Berlin, Germany, pp. 194–208, 2012.

Amorim, W. P., Falcão, A. X., Papa, J. P., and de Carvalho, M. H. Improving semi-supervised learning through optimum connectivity. *Pattern Recognition* vol. 60, pp. 72–85, 2016.

Barioni, M. C. N., Razente, H., Marcelino, A., Traina, A. J. M., and Traina-Jr, C. Open issues for partitioning clustering methods: An overview. *Wiley Int. Rev. Data Mining and Knowledge Discovery* 4 (3): 161–177, 2014.

Basu, S., Davidson, I., and Wagstaff, K. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman & Hall/CRC, 2008.

Borg, I. and Groenen, P. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

Buza, K. Semi-supervised naive hubness bayesian k-nearest neighbor for gene expression data. In *International Conference on Computer Recognition Systems*. Wroclaw, Poland, pp. 101–110, 2016.

Chapelle, O., Schlkopf, B., and Zien, A. *Semi-Supervised Learning*. The MIT Press, 2010.

Chen, J., Saad, Y., and Dasgupta, S. Fast approximate knn graph construction for high dimensional data via recursive lanczos bisection. *Jornal of Machine Learning Research* 10 (1): 1989–2012, 2009.

Databasesports.com. Database basketball. `www.databasebasketball.com`, 2011. Accessed: March, 2016.

Davidson, I. and Basu, S. A survey of clustering instance level. *Transactions on Knowledge Discovery from Data* 1 (1): 1–41, 2007.

Demsar, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 7 (1): 1–30, 2006.

Dhillon, I. S., Guan, Y., and Kulis, B. Kernel k-means: Spectral clustering and normalized cuts. In *International Conference on Knowledge Discovery and Data Mining*. Seattle, WA, pp. 551–556, 2004.

Dubey, A., Bhattacharya, I., and Godbole, S. A cluster-level semi-supervision model for interactive clustering. In *European Conference on Machine Learning and Knowledge Discovery in Databases*. Barcelona, Spain, pp. 409–424, 2010.

Faloutsos, C. and Lin, K.-I. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *ACM SIGMOD Record* 24 (2): 163–174, 1995.

Flexer, A. and Schnitzer, D. Can shared nearest neighbors reduce hubness in high-dimensional spaces? In *International Conference on Data Mining Workshops*. Dallas, Texas, pp. 460–467, 2013.

Hair Jr., J. F., Anderson, R. E., Tatham, R. L., and Black, W. C. *Multivariate Data Analysis*. Prentice-Hall, Inc., 1995.

Hubert, L. and Arabie, P. Comparing partitions. *Journal of Classification* 2 (1): 193–218, 1985.

Kailing, K. and H.P. Kriegel, P. K. Density-connected subspace clustering for high-dimensional data. In *SIAM International Conference on Data Mining*. Lake Buena Vista, Florida, pp. 246–257, 2004.

Kailing, K., H.P. Kriegel, P. K., and Wanka, S. Ranking interesting subspaces for clustering high dimensional data. In *European Conference on Principles and Practice of Knowledge Discovery in Databases*. Cavtat/Dubrovnik,Croatia, pp. 241–252, 2003.

Kira, K. and Rendell, L. A. The feature selection problem: Traditional methods and a new algorithm. In *National Conference on Artificial Intelligence*. San Jose, California, pp. 129–134, 1992.

Kullback, S. and Leibler, R. A. On information and sufficiency. *Annals of Mathematical Statistics* 22 (1): 79–86, 1951.

Kumar, N. and Kummamuru, K. Semisupervised clustering with metric learning using relative comparisons. *IEEE Transactions on Knowledge and Data Engineering* 20 (4): 496–503, 2008.

Li, J., Sander, J., Campello, R., and Zimek, A. Active learning strategies for semi-supervised DBSCAN. In *Canadian Conference on Artificial Intelligence*. Montreal, Canada, pp. 179–190, 2014.

Lichman, M. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2013. Accessed: September, 2016.

Lima, M., Barioni, M., and Razente, H. Combinando semi-supervisão e hubness para aprimorar o agrupamento de dados em alta dimensão. In *Simpósio Brasileiro de Banco de Dados*. Salvador, Brazil, pp. 139–144, 2016.

Muller, E., Gunnemann, S., Assent, I., and Seidl, T. Evaluating clustering in subspace projections of high dimensional data. *Very Large Data Base Endowment* 2 (1): 1270–1281, 2009.

Saito, P. T. M., Amorim, W. P., Falcão, A. X., de Rezende, P. J., Suzuki, C. T. N., Gomes, J. F., and de Carvalho, M. H. Active semi-supervised learning using optimum-path forest. In *Int'l Conference on Pattern Recognition (ICPR)*. IEEE, Stockholm, Sweden, pp. 3798–3803, 2014.

Samet, H. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2006.

Sander, J., Ester, M., Kriegel, H.-P., and Xu, X. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery* 2 (2): 169–194, 1998.

Satuluri, V. and Parthasarathy, S. Bayesian locality sensitive hashing for fast similarity search. *Very Large Data Base Endowment* 5 (5): 430–441, 2012.

Schmidt, J., Brandle, E. M., and Kramer, S. Clustering with attribute-level constraints. In *International Conference on Data Mining*. Vancouver, Canada, pp. 1206–1211, 2011.

Taniar, D. and Iwan, L. H. *Exploring Advances in Interdisciplinary Data Mining and Analytics: New Trends*. IGI Global, 2011.

Tomasev, N. Hub miner. `https://github.com/datapoet/hubminer`, 2014. Accessed: November, 2014.

Tomasev, N. and Mladenic, D. Hub co-occurrence modeling for robust high-dimensional knn classification. In *European Conference on Machine Learning and Knowledge Discovery in Databases*. Prague, Czech Republic, pp. 643–659, 2013.

Tomasev, N. and Mladenic, D. Hubness-aware shared neighbor distances for high-dimensional k-nearest neighbor classification. *Knowledge and Information Systems* 39 (1): 89–122, 2014.

Tomasev, N., Radovanovic, M., Mladenic, D., and Ivanovic, M. The role of hubness in clustering high-dimensional data. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Shenzhen, China, pp. 183–195, 2011.

Tomasev, N., Radovanovic, M., Mladenic, D., and Ivanovic, M. The role of hubness in clustering high-dimensional data. *IEEE Transactions on Knowledge and Data Engineering* 26 (3): 739–751, 2014.

Tomasev, N., Radovanovic, M., Mladenic, D., and Ivanovic, M. *Hubness-Based Clustering of High-Dimensional Data*. Springer International Publishing, 2015.

Traina-Jr, C., Traina, A. J. M., and Faloutsos, C. Fast feature selection using fractal dimension - ten years later. *Journal of Information and Data Management* 1 (1): 17–20, 2010.

Wagstaff, K. and Cardie, C. Clustering with instance-level constraints. In *International Conference on Machine Learning*. Williamstown, Massachusetts, pp. 1103–1110, 2000.

Wang, J., Markert, K., and Everingham, M. Learning models for object recognition from natural language descriptions. In *British Machine Vision Conference*. London, UK, pp. 1–11, 2009.

Xiong, S., Azimi, J., and Fern, X. Z. Active learning of constraints for semi-supervised clustering. *IEEE Transactions on Knowledge and Data Engineering* 26 (1): 43–54, 2014.

Zar, J. H. *Biostatistical Analysis*. Prentice-Hall, Inc., 2007.