

# Toward Autonomic Grids: Analyzing the Job Flow with Affinity Streaming

Xiangliang Zhang  
INRIA, Université Paris Sud  
LRI Bat. 490, F-91405 Orsay  
xlzhang@lri.fr

Cyril Furtlehner  
INRIA  
LRI Bat. 490, F-91405 Orsay  
Cyril.Furtlehner@lri.fr

Julien Perez  
Université Paris Sud  
LRI Bat. 490, F-91405 Orsay  
perez@lri.fr

Cecile Germain-Renaud  
Université Paris Sud  
LRI Bat. 490, F-91405 Orsay  
cecile@lri.fr

Michèle Sebag  
CNRS  
LRI Bat. 490, F-91405 Orsay  
sebag@lri.fr

## ABSTRACT

The *Affinity Propagation* (AP) clustering algorithm proposed by Frey and Dueck (2007) provides an understandable, nearly optimal summary of a dataset, albeit with quadratic computational complexity. This paper, motivated by Autonomic Computing, extends AP to the data streaming framework. Firstly a hierarchical strategy is used to reduce the complexity to  $\mathcal{O}(N^{1+\epsilon})$ ; the distortion loss incurred is analyzed in relation with the dimension of the data items. Secondly, a coupling with a change detection test is used to cope with non-stationary data distribution, and rebuild the model as needed. The presented approach STRAP is applied to the stream of jobs submitted to the EGEE Grid, providing an understandable description of the job flow and enabling the system administrator to spot online some sources of failures.

## Categories and Subject Descriptors

H.2.8 [Database Management]: Database Applications—*Data Mining*; I.5.3 [Pattern Recognition]: Clustering—*Algorithms, Similarity measures*

## General Terms

Algorithms, Experimentation

## Keywords

Affinity Propagation, Autonomic Computing, Online Clustering

## 1. INTRODUCTION

The clustering of large-scale dynamic datasets is a key issue for most application domains, at the crossroad of data-

bases, data mining and machine learning [3]. High performance computers and large-size memory storage do not *per se* sustain scalable and accurate clustering. Typically, advances in large-scale clustering (see e.g., [12]) mainly proceed by distributing the dataset and processing the subsets in parallel; when dealing with dynamic datasets, such Divide-and-Conquer approaches face some limitations in terms of latency and/or communication costs.

Furthermore, the choice of a clustering method must reflect the applicative needs. Our motivating application pertains to the strategic field of Autonomic Computing [19], aimed at providing large computational systems with self-modelling, self-configuring, self-healing and self-optimizing facilities. More specifically, the applicative goal of the present paper is to enable the administrator of a large-scale grid system, the EGEE Grid<sup>1</sup>, to analyze the flow of jobs submitted to and processed by the grid. The input data thus is made of the Logging and Bookkeeping (L&B) files, automatically generated by the grid middleware. As noted by [7], modern data mining is more and more concerned with automatically generated datasets (“computers are fueling each other”); building understandable summaries thereof is even more critical. For this reason, it is highly desirable that a job cluster be summarized by an actual job (as opposed to an artefact, as done in *K*-means; more in Section 4).

Affinity Propagation (AP), a message passing-based clustering algorithm proposed by Frey and Dueck [6], does satisfy the above interpretability constraint. Akin *K*-centers, AP maps each data item onto an actual data item, called *exemplar*, and all items mapped onto the same exemplar form one cluster. Contrasting with *K*-centers, AP builds quasi-optimal clusters in terms of distortion (section 2.1), thus enforcing the cluster stability [6]. The price to pay for these understandability and stability properties is AP quadratic computational complexity, severely hindering its usage on large scale datasets.

In an earlier work [22], a 2-level hierarchical approach was proposed to decrease AP complexity from  $\mathcal{O}(N^2)$  to  $\mathcal{O}(N^{3/2})$ , where  $N$  denotes the number of items. Independently, some coupling with a change detection test was in-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'09, June 28–July 1, 2009, Paris, France.

Copyright 2009 ACM 978-1-60558-495-9/09/06 ...\$5.00.

<sup>1</sup>The EGEE grid was established in the EU project *Enabling Grid for E-Science*, <http://www.eu-egee.org>. It involves 41,000 CPUs, 5 Petabytes storage and concurrently supports 20,000 jobs on 24/24, 7/7 basis.

investigated to extend AP to dynamic data distributions, enabling online clustering aka Data Streaming [22].

This paper features three contributions along these same lines. Firstly, a straightforward generalization of the 2-level hierarchical approach is proposed, showing that a  $h$ -level hierarchical approach would reduce the computational complexity to  $\mathcal{O}(N^{\frac{h+2}{h+1}})$  up to poly-logarithmic terms. Secondly, the price to pay for this complexity reduction, namely the distortion loss incurred along the hierarchical Divide-and-Conquer, is analyzed; the distortion loss is shown to be negligible except in the particular case of two-dimensional datasets. Thirdly, an adaptive mechanism inspired from [21] is used to optimize the change-detection test parameters.

The extended STRAP algorithm (*Streaming Affinity Propagation*) is finally applied to a challenging real-world problem, the online monitoring of the EGEE grid. The STRAP specificity compared to prominent data streaming algorithms (e.g., [2, 3, 7]) is twofold. On the one hand, STRAP inherits AP understandability, modelling the data stream through exemplars (actual data items) as opposed to artefacts. On the other hand, this model is available at any time step contrasting with e.g., [2]. STRAP thus makes it feasible to provide the EGEE administrator with a real-time dashboard of the job data flow, enabling the discovery of anomalies.

The paper is organized as follows. Section 2 first briefly describes AP for the sake of self-containedness, before presenting Divide-and-Conquer AP. The computational complexity thereof is derived, and the distortion loss is analyzed. The STRAP algorithm extending Hi-AP to data-streaming is presented in Section 3, and the self-adaptive change detection test is detailed. Section 4 describes a large scale real-world application: the profiling of 5M+ jobs submitted to the EGEE grid. STRAP is assessed in terms of algorithmic robustness and performance compared to a  $k$ -centers baseline approach, and the added value for the grid administrator is discussed. The paper concludes with some perspectives for further research.

## 2. SCALABLE CLUSTERING WITH AP

This section presents the Affinity Propagation algorithm, referring the reader to [6] for a comprehensive description. How to make AP scalable, and what is the price to pay for the complexity reduction, are described thereafter.

### 2.1 AP and Weighted AP

Let  $\mathcal{E} = \{e_1, \dots, e_N\}$  denote a set of  $N$  items, and let  $d(e_i, e_j)$  denote the distance or dissimilarity between items  $e_i$  and  $e_j$ . Letting  $K$  denote a positive integer, the  $K$ -center problem consists of finding  $K$  items  $e_{i_1}, \dots, e_{i_K}$  in  $\mathcal{E}$ , referred to as exemplars, minimizing the dataset *distortion*  $\mathcal{L}$ :

$$\mathcal{L} = \sum_{j \in 1 \dots N} \min_{k \in 1 \dots K} d^2(e_j, e_{i_k})$$

defined as the squared distance between  $e_j$  and its closest exemplar, summed over all  $e_j$  in  $\mathcal{E}$ .

Affinity Propagation is a message-passing algorithm tackling the above optimization problem as follows. Let  $\mathbf{c}$  be defined as a mapping from  $\mathcal{E}$  onto  $\mathcal{E}$ , associating to each item  $e_i$  its exemplar  $c_i$  ( $c_i \in \mathcal{E}$ ).

$$\text{Find } \mathbf{c}^* = \text{argmax}(E[\mathbf{c}]),$$

with

$$E[\mathbf{c}] = \sum_{i=1}^N S(e_i, c_i) - \sum_{i=1}^N \log \chi_i^{(p)}[\mathbf{c}] \quad (1)$$

where

$$S(e_i, c_i) = \begin{cases} -d^2(e_i, c_i) & \text{if } e_i \neq c_i \\ -s_* & \text{otherwise} \end{cases}$$

Parameter  $s_*$ , the penalty for having an additional exemplar, controls the clustering granularity:  $s_* = 0$  leads to the trivial solution where every item is an exemplar;  $s_* = \infty$  leads to the single-cluster solution. Contrasting with  $K$ -centers, AP only indirectly controls the number of clusters through  $s_*$  (usually set to the median value of  $d^2(e_i, e_j)$ ).  $\chi_i^{(p)}[\mathbf{c}]$  is a set of constraints initially meant to enforce the fact that, if  $e_i$  is chosen as exemplar, it must be its own exemplar (clusters are ball-shaped) [6]. A relaxation of the  $\chi$  constraints enabling tree-structured clusters, was proposed by [13].

The Divide-and-Conquer approach presented next relies on the Weighted AP algorithm (WAP) [22], extending AP to the case of multiply-defined items, and/or dense subsets of items. Let  $\mathcal{F}$  denote a subset of  $\mathcal{E}$ , made of  $n$  items with small pair distance ( $\forall e_i, e_j \in \mathcal{F}, d^2(e_i, e_j) < \epsilon$ ). WAP proceeds by replacing all items in  $\mathcal{F}$  with a single example  $e_f$ . The clustering problem defined on  $\mathcal{E}$  is made equivalent to the one defined on  $(\mathcal{E} \setminus \mathcal{F}) \cup \{e_f\}$ , by setting:

$$\begin{aligned} S(e_f, e_j) &\longrightarrow \sum_{e_i \in \mathcal{F}} S(e_i, e_j), & \forall e_j \in \mathcal{E} \setminus \mathcal{F} \\ S(e_j, e_f) &\longrightarrow \frac{1}{n} \sum_{e_i \in \mathcal{F}} S(e_j, e_i), & \forall e_j \in \mathcal{E} \setminus \mathcal{F} \\ S(e_f, e_f) &\longrightarrow -s_* + (n-1)\epsilon \end{aligned}$$

The WAP algorithm will be used in the remainder of the paper to iteratively cluster exemplars produced in former clustering steps.

### 2.2 Complexity of Hierarchical AP

AP computational complexity<sup>2</sup> is  $N^2 \log(N)$ ; it involves the matrix  $S$  of pair distances, with quadratic complexity in the number  $N$  of items, severely hindering its use on large-scale datasets.

This AP limitation can be overcome through a Divide-and-Conquer heuristics inspired from [9]. Dataset  $\mathcal{E}$  is randomly split into  $b$  data subsets; AP is launched on every subset and outputs a set of exemplars; the exemplar weight is set to the number of initial samples it represents; finally, all weighted exemplars are gathered and clustered using WAP (the complexity is  $\mathcal{O}(N^{3/2})$  [22]). This Divide-and-Conquer strategy – which could actually be combined with any other basic clustering algorithm – can be pursued hierarchically in a self-similar way, as a branching process with  $b$  representing the branching coefficient of the procedure, defining the Hierarchical AP (Hi-AP) algorithm.

Formally, let us define a tree of clustering operations, where the number  $h$  of successive random partitions of the data represents the height of the tree. At each level of the hierarchy, the penalty parameter  $s_*$  is set such that the expected number of exemplars extracted along each clustering step is upper bounded by a constant  $K$ .

<sup>2</sup>Except if the similarity matrix is sparse, in which case the complexity reduces to  $NK \log(N)$  with  $K$  the average connectivity of the  $S$  matrix [6].

PROPOSITION 2.1. Let us define the branching factor  $b$  as

$$b = \left(\frac{N}{K}\right)^{\frac{1}{h+1}},$$

Then the overall complexity  $C(h)$  of HI-AP is given by

$$C(h) \propto K^{\frac{h}{h+1}} N^{\frac{h+2}{h+1}} \quad N \gg K,$$

up to logarithmic terms.

PROOF.  $M = N/b^h$  is the size of each subset to be clustered at level  $h$ ; at level  $h - 1$ , each clustering problem thus involves  $bK = M$  exemplars with corresponding complexity

$$C(0) = K^2 \left(\frac{N}{K}\right)^{\frac{2}{h+1}}.$$

The total number  $N_{cp}$  of clustering procedures involved is

$$N_{cp} = \sum_{i=0}^h b^i = \frac{b^{h+1} - 1}{b - 1},$$

with overall computational complexity:

$$C(h) = K^2 \left(\frac{N}{K}\right)^{\frac{2}{h+1}} \frac{\frac{N}{K} - 1}{\left(\frac{N}{K}\right)^{\frac{1}{h+1}} - 1} \underset{N \gg K}{\approx} K^2 \left(\frac{N}{K}\right)^{\frac{h+2}{h+1}}.$$

It is seen that  $C(0) = N^2$ ,  $C(1) \propto N^{3/2}$ ,  $\dots$ , and  $C(h) \propto N$  for  $h \gg 1$ .  $\square$

### 2.3 Distortion Regret of HI-AP

Let us examine the price to pay for this complexity reduction. As mentioned earlier on, the clustering quality is assessed from its distortion, the sum of the squared distance between every data item and its exemplar:

$$D(\mathbf{c}) = \sum_{i=1}^N d^2(e_i, c_i)$$

The distortion loss incurred by HI-AP w.r.t. AP is examined in the simple case where the data samples follow a centered distribution in  $\mathbb{R}^d$ . By construction, AP aims at finding the cluster exemplar  $\mathbf{r}_{\mathbf{c}}$  nearest to the center of mass of the sample points noted  $\mathbf{r}_{cm}$ :

$$D(\mathbf{c}) = |\mathbf{r}_{cm} - \mathbf{r}_{\mathbf{c}}|^2 + Cst$$

The distortion loss incurred by HI-AP can be assessed from the relative entropy, or Kullback Leibler distance, between the distribution  $P_{\mathbf{c}}$  of the cluster exemplar computed by AP, and the distribution  $P_{\mathbf{c}(h)}$  of the cluster exemplar computed by HI-AP with hierarchy-depth  $h$ :

$$D_{KL}(P_{\mathbf{c}}||P_{\mathbf{c}(h)}) = \int P_{\mathbf{c}(h)}(r) \log \frac{P_{\mathbf{c}(h)}(r)}{P_{\mathbf{c}}(r)} dr \quad (2)$$

In the simple case where points are sampled along a centered distribution in  $\mathbb{R}^d$ , let  $\tilde{\mathbf{r}}_{\mathbf{c}}$  denote the relative position of exemplar  $\mathbf{r}_{\mathbf{c}}$  with respect to the center of mass  $\mathbf{r}_{cm}$ :

$$\tilde{\mathbf{r}}_{\mathbf{c}} = \mathbf{r}_{\mathbf{c}} - \mathbf{r}_{cm}$$

The probability distribution of  $\tilde{\mathbf{r}}_{\mathbf{c}}$  conditionally to  $\mathbf{r}_{cm}$  is cylindrical; the cylinder axis supports the segment  $(0, \mathbf{r}_{cm})$ , where 0 is the origin of the  $d$ -dimensional space. As a result, the probability distribution of  $\mathbf{r}_{cm} + \tilde{\mathbf{r}}_{\mathbf{c}}$  is the convolution of a spherical with a cylindrical distribution.

Let us define the following notations. Subscripts  $sd$  refer to sample data,  $ex$  to the exemplar, and  $cm$  to center of

mass. Let  $x$ , denote the corresponding square distances to the origin,  $f$ , the corresponding probability densities and  $F$ , their cumulative distribution. Assuming

$$\sigma \stackrel{\text{def}}{=} \mathbb{E}[x_{sd}] = \int_0^\infty x f_{sd}(x) dx,$$

and

$$\alpha \stackrel{\text{def}}{=} - \lim_{x \rightarrow 0} \frac{\log(F_{sd}(x))}{x^{\frac{d}{2}}},$$

exist and are finite, then the cumulative distribution of  $x_{cm}$  of a sample of size  $M$  satisfies

$$\lim_{M \rightarrow \infty} F_{cm}\left(\frac{x}{M}\right) = \frac{\Gamma\left(\frac{d}{2}, \frac{2x}{d\sigma}\right)}{\Gamma\left(\frac{d}{2}\right)}.$$

by virtue of the central limit theorem. In the meanwhile,  $x_{\tilde{ex}} = x_{ex} - x_{cm}$  has a universal extreme value distribution (up to rescaling):

$$\lim_{M \rightarrow \infty} F_{\tilde{ex}}\left(\frac{1}{M^{2/d}}x\right) = \exp(-\tilde{\alpha}x^{\frac{d}{2}}).$$

where  $\tilde{\alpha} \neq \alpha$  stands for the fact that the extreme value parameter is possibly affected by the displacement of the center of mass. To see how the clustering error propagates along with the hierarchical process, one proceeds inductively. At hierarchical level  $h$ ,  $M$  samples, spherically distributed with variance  $\sigma^{(h)}$  are considered; the sample nearest to the center of mass is selected as exemplar. Accordingly, at hierarchical level  $h + 1$ , the next sample data is distributed after the convolution of two spherical distributions, the exemplar and center of mass distributions at level  $h$ . The following scaling recurrence property (proof in appendix) holds:

PROPOSITION 2.2.

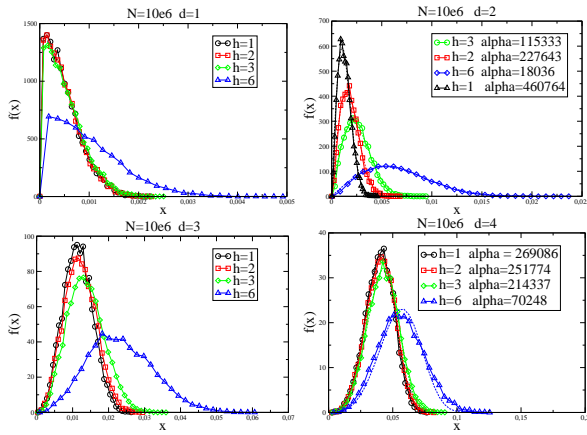
$$\lim_{M \rightarrow \infty} F_{sd}^{(h+1)}\left(\frac{x}{M^{(h+1)\gamma}}\right) = \begin{cases} \frac{\Gamma\left(\frac{d}{2}, \frac{x}{\sigma^{(h+1)}}\right)}{\Gamma\left(\frac{d}{2}\right)} & d < 2, \gamma = 1 \\ \exp(-\alpha^{(h+1)}x^{\frac{d}{2}}) & d > 2, \gamma = \frac{2}{d} \\ \exp(-\beta^{(h+1)}x) & d = 2, \gamma = 1. \end{cases}$$

with

$$\sigma^{(h+1)} = \sigma^{(h)}, \quad \alpha^{(h+1)} = \alpha^{(h)}, \quad \beta^{(h+1)} = \frac{\beta^{(h)}}{2}.$$

It follows that the distortion loss incurred by HI-AP does not depend on the hierarchy depth  $h$  except in dimension  $d = 2$ .

Fig. 1 shows the distribution of the clustering distortion depending on the hierarchy-depth  $h$  and the dimension  $d$  of the dataset. The distortion curve for  $h = 1$  corresponds to the AP case, showing that the distortion loss due to the hierarchical approach is moderate to negligible in dimension  $d \neq 2$  provided that the number of samples per cluster at each clustering level is "sufficient" (say,  $M > 30$  for the law of large numbers to hold). In dimension  $d > 2$ , the distance of the center of mass to the origin is negligible with respect to its distance to the nearest exemplar; the distortion behaviour thus is given by the Weibull distribution which is stable by definition (with an increased sensitivity to small sample size  $M$  as  $d$  goes to 2). In dimension  $d = 1$ , the distribution is dominated by the variance of the center of mass, yielding the gamma law which is also stable with respect to



**Figure 1: Radial distribution plot of exemplars obtained by clustering of Gaussian distributions of  $N = 10^6$  samples in  $\mathbb{R}^d$  in one single cluster exemplar, with hierarchical level  $h$  ranging in 1,2,3,6, for diverse values of  $d$ :  $d = 1$  (upper left),  $d = 2$  (upper right),  $d = 3$  (bottom left) and  $d = 4$  (bottom right). Fitting functions are of the form  $f(x) = Cx^{d/2-1} \exp(-\alpha x^{d/2})$ .**

the hierarchical procedure. In dimension  $d = 2$  however, the Weibull and gamma laws do mix at the same scale; the overall effect is that the width of the distribution (of the distortion) increases like  $h^2$ , as shown in Fig. 1 (top right).

### 3. MODELING DATA STREAMS WITH AP

This section is concerned with adapting HI-AP to online clustering and dynamic data distributions, defining the STRAP algorithm. STRAP combines HI-AP with a change detection test: if the test is triggered, it is likely that the data distribution has changed and the stream model is rebuilt. Two extensions have been brought to STRAP initial version [22]. An adaptive mechanism inspired from [21] is used to automatically optimize the parameters of the change detection test. Secondly, the exemplars built by STRAP fuel an offline clustering process, enabling some multi-scale description of the data stream. This section finally discusses the strengths and weaknesses of STRAP with respect to the state of the art.

#### 3.1 Exemplar-based Clustering with Change Detection

The early STRAP algorithm is summarized for the sake of self-containedness, referring the reader to [22] for more detail. The model of the stream is initialized by applying HI-AP to the first data items. Formally, the stream model is made of a set of clusters  $C_i = (e_i, n_i, \Sigma_i, t_i)$ , where  $e_i$  is the cluster exemplar,  $n_i$  and  $\Sigma_i$  respectively stand for the cluster size and distortion, and  $t_i$  is the last time stamp when a data item joined the cluster.

As the stream flows in, current data item  $e_t$  is checked against the model. If its distance to the nearest exemplar  $e_i$  is less than a threshold computed in the initialization step,  $e_t$  joins the  $C_i$  cluster. The  $C_i$  time stamp is set to the current time step  $t$ , while  $C_i$  size and distortion are updated

by relaxation. The model update is parameterized from a (user supplied) time length  $\Delta$ ; the idea is that clusters which have not received any additional item during  $\Delta$  consecutive time steps [22] should disappear. If data item  $e_t$  does not fit the model, it is considered to be an outlier and put in the reservoir. The reservoir gathers the last  $M$  outliers.

A change point detection test is used to monitor the stability of the data distribution. The so-called Page Hinkley (PH) statistical test [16, 11] is applied to the outlier rate (section 3.2). Upon triggering the PH test, the stream model is rebuilt using WAP from the current model (exemplars weighted by the current size of the associated cluster) and the outliers in the reservoir.

---

#### Algorithm 1 STRAP Algorithm

---

**Data streams**  $e_1, \dots, e_t, \dots$ ; **fit threshold**  $\varepsilon$

**Init**

AP( $e_1, \dots, e_T$ )  $\rightarrow$  STRAP Model

Reservoir =  $\{\}$

**for**  $t > T$  **do**

  Compute  $e_i =$  nearest exemplar to  $e_t$

**if**  $d(e_t, e_i) < \varepsilon$  **then**

    Update STRAP model

**else**

    Reservoir  $\leftarrow e_t$

**end if**

**if**  $PH_t > \lambda$  **then**

    Rebuild STRAP model

    Reservoir =  $\{\}$

**end if**

**end for**

---

#### 3.2 Self-Adaptive Change Detection Test

Among the main change detection tests are Wald tests, also referred to as SPRT (sequential probability ratio test [15]), and CUSUM test (cumulative sum [16]); kernelized versions of the CUSUM test have also been developed, see e.g., [10]. The Page-Hinkley test has been used within the STRAP algorithm because it minimizes the time expectancy before detecting a change, conditionally to a given false alarm rate [16, 11].

Formally, the PH test monitors a scalar random variable  $p_t$ . The test is parameterized after a threshold  $\lambda$ , classically governing the rate of false alarms; additionally, a small constant tolerance parameter  $\delta$  has been used to cope with slowly varying distributions:

$$\bar{p}_t = \frac{1}{t} \sum_{\ell=1}^t p_\ell \quad m_t = \sum_{\ell=1}^t (p_\ell - \bar{p}_\ell + \delta)$$

$$M_t = \max\{|m_\ell|, \ell = 1..t\} \quad PH_t = (M_t - |m_t|) > \lambda$$

In its current state, the PH test is only triggered when  $p_t$  tends to increase, as the monitored variable  $p_t$  relates to the presence of outliers. When the rate or severity of outliers decreases, there is no need to rebuild the stream model. Several scalar indicators  $p_t$  have been considered, among which the distance of the current data item to the nearest exemplar, possibly normalized by the associated distortion. Empirically, the best performing indicator is found to be

$$p_t = \frac{1 + \mathbf{1}_{O_t}}{1 + O_t}$$

where  $\mathbf{1}_{o_t}$  is set to 1 if the current data item is an outlier and 0 otherwise, and  $O_t$  is the fraction of data items considered to be outliers since the model was last (re)built.

Threshold  $\lambda$  is adjusted in order to optimize the model representativity, in the spirit of the Bayesian Information Criterion [20]. The optimization criterion is set to:

$$\mathcal{F}_\lambda = -\frac{1}{|C|} \sum_{i=1}^{|C|} \left( \frac{1}{n_i} \sum_{e_j \in C_i} d(e_j, e_i^*) \right) - \varphi \frac{d}{2} \log N - \eta O_t \quad (3)$$

where  $|C|$  is the number of clusters,  $e_i^*$  and  $n_i$  respectively the exemplar and size of the  $i$ -th cluster,  $d$  the dimension of the data stream,  $N$  the number of data items recognized by the stream model since the last restart,  $\varphi$  and  $\eta$  are two constants to make the penalty term on the same scale as the distortion item.

The optimization of  $\lambda$  has been tackled in a discrete (considering a finite set of values) and a continuous (considering a continuous domain) setting, respectively using  $\epsilon$ -greedy optimization and a Gaussian Process-based estimate of  $\mathcal{F}_\lambda$  [21].

### 3.3 Multi-Scale Modeling of the Stream

While data streaming aims at providing an accurate description of the instant flow distribution, it might be desirable to also provide “the big picture”, depicting the evolution of this distribution on a larger time scale.

The fact that at each time step the stream model is based on exemplars makes it natural to apply Hi-AP on the overall set of exemplars gathered along time<sup>3</sup>, thus extracting “super-exemplars”. These super-exemplars capture the various trends of the data stream along time, enabling to characterize any period (day, week or month) after the representativity of each such super-exemplar (number of data items falling in each super-cluster).

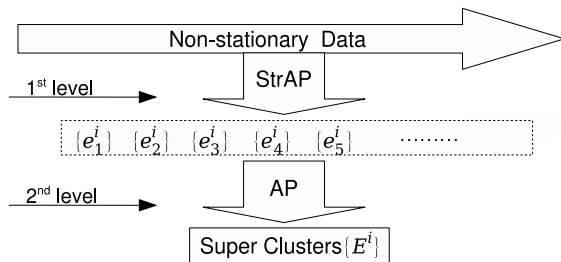


Figure 2: Online (1st level) and retrospective (2nd level) representation of a data stream with STRAP.

### 3.4 Discussion

Among the prominent challenges of Data Streaming (see e.g., [8, 3]) are the computational time and space resources needed, on the one hand, and the efficient modeling of non-stationary distributions on the other hand. There is little doubt that the computational requirements of a data streaming algorithm govern its usability; typically when the system under examination generates the equivalent of 27 CDs per minute, linear or quasi-linear computational complexity is the maximum one can afford to keep up with real-time processing. The second challenge, namely the pursuit

<sup>3</sup>The exemplars with low representativity are filtered out.

of a moving target distribution, has been extensively considered in the Signal Processing and Data Analysis literature [1]. It however needs to be reconsidered in the Data Streaming context, subject to the above mentioned computational limitations. This challenge can be viewed as yet another instance of the Exploration vs Exploitation dilemma; indeed, a competent data streaming algorithm should simultaneously be able to catch up with any true change in the data distribution (exploration) while discarding outliers in order not to spoil the model (exploitation).

A third and equally important challenge is to provide the user with understandable results. As ill-defined as understandability might be, it remains that providing understandable results is mandatory in order to keep the user in the loop [5]. The presented STRAP algorithm aims at understandable, stable and computationally efficient Data Streaming, through the selection of the exemplars best representing the (majority of) data items at any time step. The Continuous Distributed clustering (CDC) algorithm presented by Cormode et al. [3] is most related to STRAP, with two important differences. Firstly, CDC is interested in “conquering the divide”, i.e. building a model of a distributed stream, whereas STRAP is interested in splitting the data stream to overcome the complexity barrier. Secondly, CDC is based on  $K$ -centers and its goal is to minimize the radius (maximal distance between a point and its exemplar) or the diameter (maximal distance between two points with same exemplars) of the clustering, whereas the STRAP goal is to minimize the distortion; the difference between both criteria can be understood as the difference between  $L_\infty$  and  $L_1$  or  $L_2$  norms.

To our best knowledge, STRAP is the only Data Streaming algorithm modeling a centralized data flow through a set of exemplars. This unique feature of STRAP is both a strength and a weakness compared to the Data Streaming algorithms at the state of the art. On the weak side, STRAP was shown to be slower by an order of magnitude than *DenStream* [2] on the KDD Intrusion Detection Dataset [22]. This lesser computational efficiency is blamed on two facts. Firstly, *DenStream*, extending the DBScan clustering algorithm [4] to the streaming context, constructs an artefact-based model, smoothly updating an implicit  $K$ -means-like model at any time step, whereas STRAP explicitly rebuilds the model whenever some change in the underlying data distribution is detected. Secondly, the data model is available at any time step in STRAP, whereas it is only computed upon request by *DenStream*.

In counterpart, to our best knowledge STRAP is the only applicable Data Streaming algorithm when i) the data representation makes it impossible to build artefacts; ii) some performance and stability/reproducibility guarantees are needed, barring the use of  $K$ -centers. Domains such as molecular chemistry, image processing, or social networks fall in the first category (e.g., defining an “average” molecule still is an open problem). Safety-critical domains fall in the second category.

## 4. MONITORING THE JOB FLOW IN A GRID SYSTEM

This section reports on the application of STRAP to the Autonomic Grid context, specifically the monitoring of the jobs submitted to the EGEE grid. After briefly describing the application, the section describes the goal of experiments

and the experimental setting, before discussing the empirical results.

## 4.1 Grid Monitoring and Job Streaming

Grid Monitoring involves two main functionalities: acquisition and usage of the relevant information. The acquisition functionality includes sensors that instrument grid services or applications, and data collection services that filter, centralize and/or distribute the sensor data to the usage functionality. Usage, which is more specifically investigated in this paper, includes consumer services such as real-time presentation and interpretation. It also includes middleware services as far as feedback loops are considered, typically in the Autonomic Computing framework. Many architectures and integration frameworks offer advanced presentation, user interaction and reporting facilities, such as the EGEE DASHBOARD [23] and Real Time Monitor [24]. Data interpretation, aimed at revealing meaningful (compound) features which go beyond elementary statistics, is much less developed in the grid area.

The goal of the proposed Job Streaming facility, enabling the real-time inspection of the jobs submitted to and processed by the grid, is to provide some interpretation of the grid running status. The job stream considered in the following is the log of 39 Resource Breakers (RB) of all gLite-operated jobs in the whole EGEE grid from early January 2006 to end of May, including a total of 5,268,564 jobs. Through the Real Time Monitor system (RTM) [24], the acquisition module provides a real-time description of the jobs through XML records (available at <http://www.grid-observatory.org/>). Each job is labeled after its final status, successfully finished (*good job*) or failed. Circa 45 error classes (e.g., “Cancel requested by WorkloadManager”, “RB Cannot plan”) exist; about 25 error classes are significantly represented (with more than 1,500 occurrences) in the job stream. These labels will *not* be accounted for in the clustering process, for the following reason. Following grid experts, error types do not necessarily relate to operational aspects and could blur the picture of the grid status. For instance, although *cannot plan* means that the Resource Broker was unable to find a matching resource, the real cause might be that the user’s requests were truly unreachable; or the Broker information is stalled and does not see that resources have been released. Therefore, the job labels will only be used *a posteriori* to assess the clustering performance.

Each job is described by 6 continuous and 6 boolean attributes<sup>4</sup>. The first 6 attributes describe the time cost duration spent in different services along the job lifecycle:

1. *Submission\_Time*: time for submission to Workload Management System (WMS)
2. *Waiting\_Time*: time to find a matching resource
3. *Ready\_for\_Transfer\_Time*: time acceptance and transfer to the found resource, reported by JobController (JC)
4. *Ready\_for\_CE\_accept\_Time*: the same as Ready\_for\_Transfer\_Time, but reported by LogMonitor (LM)
5. *Scheduled\_Time*: queuing delay in local cite
6. *Running\_Time*: execution time.

<sup>4</sup>An additional categorical attribute, the name of the queue visited by the job will not be considered in this paper, although a proper handling of categorical attributes was the main motivation for using exemplar-based clustering as opposed to *K – means* approaches.

In principle, attributes 3 and 4 are redundant (JC is a standalone logging service, while the LM integrates various logs, and returns them in the L&B database); as will be seen, the discrepancy between both attributes however provide useful clues about grid misbehaviors. All numerical attributes are centered and normalized; the first data subset is used to estimate the average and standard deviation, which are thereafter updated using an additive relaxation scheme as the dataflow goes in.

In case the job does not reach a given service due to a failure in the job lifecycle, the durations of all subsequent services are set to 0. Six additional boolean attributes are thus considered, indicating whether the job reaches the corresponding service. The job dissimilarity is the Euclidean distance on  $\mathbb{R}^{12}$ .

## 4.2 Experiment Goal and Setting

The goal of the experiments is to assess the STRAP algorithm from an algorithmic and an applicative perspectives. On the one hand, STRAP is assessed from its ability to provide useful hints on the grid state. On the other hand, the algorithmic performance of STRAP is assessed with comparison to hierarchical *k*-centers streaming<sup>5</sup>, measured after four criteria:

The **Clustering Accuracy** and **Clustering Purity** are classically measured with respect to the job labels: the accuracy is the percentage of jobs with same class label as their exemplar; the purity is the fraction of the jobs in each cluster belonging to the majority class of the cluster, averaged over all clusters. The clustering purity is known to be more robust than the clustering accuracy in case of imbalanced clusters and/or classes.

The **Clustering Stability** measures the clustering performance after [14]. Specifically, a set of super-exemplars computed for a given setting  $s_*$  of STRAP induces a partition  $C(s_*)$  of the jobs; the sensitivity of the algorithm is measured from the independence of the partitions obtained for different settings, defined as the sum taken over all clusters  $C_i \in C(s_*), C'_j \in C(s'_*)$  of the quantity

$$\frac{P(e_k \in C_i \cap C'_j | e_k \in C_i) \cdot P(e_k \in C_i \cap C'_j | e_k \in C'_j)}{\min(|C(v)|, |C(v')|)}$$

The **Streaming Stability** measures the stability of the stream model w.r.t. the change detection test, and parameter  $\lambda$ . The model dynamics is reflected by the restart schedule (number of restarts per day); its stability is assessed by computing the correlation of the restart schedules obtained for various values of  $\lambda$ ; the significance of the correlation is measured after a permutation test (considering the correlation values obtained for 100 restart schedules with randomly ordered days).

In all experiments, the penalty parameter  $s_*$  of AP is initially set to the median similarity value in the first bunch of the data stream, and updated by relaxation from the sequence of data considered.

The outlier threshold is set to  $\varepsilon = 0.25$ . The PH tolerance threshold is set to  $\delta = 0.01$ . The PH threshold parameter  $\lambda$  is adjusted online, using discrete or continuous optimisation. In the discrete case,  $\lambda$  ranges in 40, 50, . . . 120 and a  $\alpha$ -greedy

<sup>5</sup>The baseline algorithm is defined by replacing AP with *k*-centers with multiple restarts; the best performance out of 30 restarts is kept, ensuring that STRAP and the baseline algorithm have same computational runtime.

optimization of the empirical average distortion (section 3.2) is achieved ( $\alpha = 5\%$ ). In the continuous case, a Gaussian Process-based estimate of the distortion (Eq. (3)) is built, the the  $\lambda$  value with minimal estimated empirical distortion is selected and the model is updated [21].

### 4.3 Algorithmic Assessment

The accuracy and purity of the STRAP modelling are respectively displayed in Fig. 3 and 4. With respect to the 21 classes of jobs, the accuracy is consistently over 85%, significantly outperforming the baseline algorithm. The adaptive adjustment of the  $\lambda$  parameter, based on discrete or continuous optimization, preserves STRAP accuracy while decreasing the number of restarts (omitted for space limitations). The clustering purity (Fig. 4) is over 90% and confirms the quality of the clustering model. The relatively high number of clusters (circa 200) must be understood in relation with the average number of jobs per day (circa 15,000). Unexpectedly, the clustering purity is higher than the accuracy, although the former indicator usually is a pessimistic one (since all clusters, including those related to rare classes, have same weight); experimentally, the difference in performance is explained as rare failure classes are associated to pure clusters.

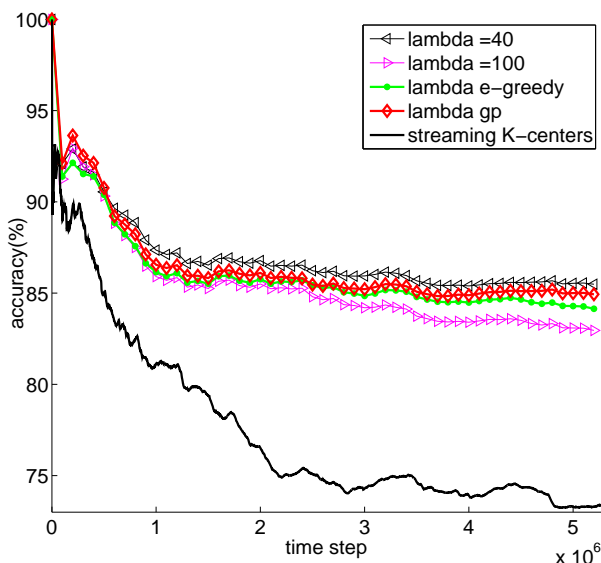


Figure 3: STRAP: Online Clustering Accuracy

The stability of the data stream model is measured considering various values of the AP penalty parameter  $s_*$ , ranging in 2.95, 3.21, 4.42, 4.82. Table 1 displays the correlation of the partitions induced by the super-exemplars (clusters with representativity less than .5% of the jobs have been filtered out): columns 3 and 4 respectively indicate the number of clusters obtained for  $s_*$  values in columns 5 and 6. The actual correlation (column 1) is assessed from the reference value (column 2), computed as the best correlation of clustering  $C_i$  with 100 random perturbations of  $C_2$ .

Finally, the stability of the model dynamics is measured by varying  $\lambda$  in 40, 50, 75 and 100, respectively inducing 699, 558, 371 and 284 restarts. The correlation between the restart schedules is significant up to the confidence level 99% (Table 2):

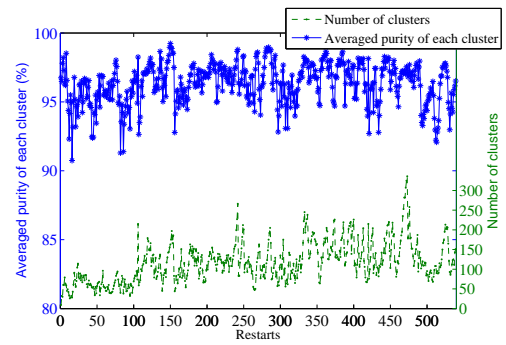


Figure 4: STRAP: Clustering Purity and Number of Clusters, measured at each restart step (discrete optimization of the change detection test parameter).

Table 1: Stability of the Data Stream Clustering Model

Correlation	Reference	$C_1$	$C_2$	$s_{*C_1}$	$s_{*C_2}$
0.7700	0.0430	62	103	-4.82	-4.42
0.7362	0.0451	79	124	-3.21	-2.95
0.7094	0.0398	79	103	-3.21	-4.42
0.6609	0.0351	62	69	-4.82	-8.84
0.6504	0.0353	79	69	-3.21	-8.84

Table 2: Correlation between the restart schedules for different  $\lambda$  values (Reference = maximal correlation out of 100 permutation tests)

$\lambda_1$	$\lambda_2$	Correlation	Reference
40	50	0.88	0.17
40	75	0.77	0.24
40	100	0.70	0.19
50	75	0.79	0.26
50	100	0.74	0.18
75	100	0.87	0.20

### 4.4 Applicative Assessment

The typical summaries of the job flow provided by STRAP to the EGEE system administrator are displayed in Fig. 5. The top snapshot corresponds to a standard situation, with a few outliers (*Reservoir*), circa 10% jobs stopping after registration (exemplar (7 0 0 0 0)), circa 15% stopping before arriving at the CE (exemplar (10 47 54 129 0 0)), about 60% successful short jobs and 10% computationally heavy jobs. Two days later (bottom snapshot), a new cluster appear including about 40% of the jobs. The corresponding exemplar (10 18 29 20091 395 276) is immediately interpreted by the administrator as an alarm signal; LM is getting clogged (exemplar value 20091s, higher than the standard one by two orders of magnitude).

The load dynamics and trends can be assessed from the number of model restarts per day (Fig. 6), comforted by the robustness analysis of this indicator presented in the previous section. This indicator however only provides a coarse feedback, for frequent restarts can be explained from several causes: i) the load is abruptly increasing; ii) new job patterns appear; iii) job patterns oscillate, frequently appearing and disappearing.

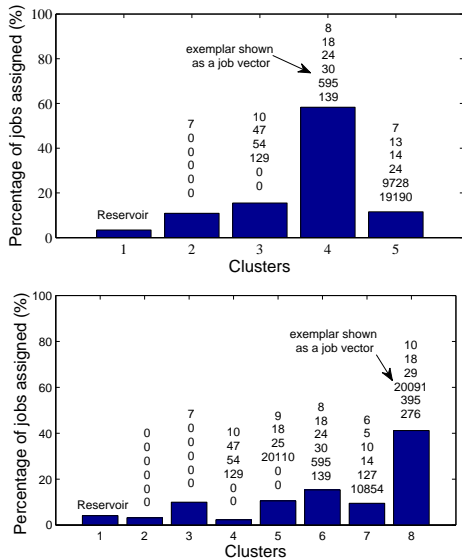


Figure 5: Online Snapshots of the Job Stream

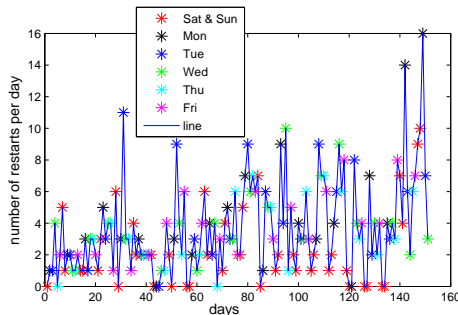


Figure 6: STRAP: Number of restarts per day

A more detailed view of the load dynamics in a long time-scale is based on using super-exemplars (section 3.3). The overall stream is visualized as a tapestry, each row corresponding to a given super-exemplar, and each column corresponding to a day (or a time period; a zooming functionality allows the administrator to adjust the granularity of the visualization). The color of the super-exemplar indicates the percentage (or number) of jobs associated to this super-exemplar in the time period, enabling the administrator to spot the load regularities.

## 5. DISCUSSION AND PERSPECTIVES

Resuming an earlier work devoted to Data Streaming with Affinity Propagation [22], this paper shows that the computational complexity of the STRAP algorithm can be reduced to a quasi-linear complexity through a generalized Divide-and-Conquer approach – without incurring a significant distortion loss except in dimension 2. Further, an adaptive procedure automatically adjusting the parameters of the change detection test has been proposed. Besides its theoretical analysis, the STRAP algorithm has been validated on a challenging real-world application, specifically the modelling of the EGEE Grid status from the job data flow. The first

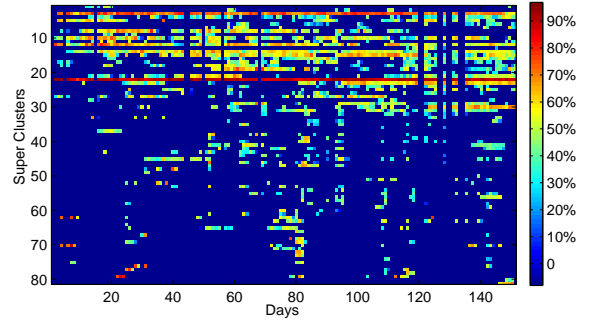


Figure 7: Visualization of the Stream Model along time (x axis: time; y axis: super-exemplars ordered by attribute 4)

results reported in this paper, made possible by the Grid Observatory initiative<sup>6</sup>, have been considered to provide relevant and useful hints into the types and hidden causes of the grid traffic jams.

A current limitation of the approach remains its computational cost; while it meets the real-time constraint of the EGEE job stream, its batch performances still are about 8 hours by Matlab code and 2 hours by C/C++ (on Intel 2.66GHz Dual-Core PC with 2 GB memory) for 5M+ jobs.

Another limitation, deeply rooted in the AP frame, is that the number of exemplars is not easily controlled from the penalty parameter  $s_*$ . How to address this limitation is our main perspective for further study. Independently, the STRAP framework will be enhanced with visual mining facilities to support a flexible multi-scale dashboard.

## Acknowledgments

This work has been partly supported by the Pascal-2 European Network of Excellence and the EGEE-III Infrastructure Project.

## 6. REFERENCES

- [1] F. Bergeaud and S. Mallat. Matching pursuit of images. In *ICIP*, pages 53–56, 1995.
- [2] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *SIAM Conference on Data Mining (SDM)*, pages 326–337, 2006.
- [3] G. Cormode, S. Muthukrishnan, and W. Zhuang. Conquering the divide: Continuous clustering of distributed data streams. In *ICDE*, pages 1036–1045, 2007.
- [4] M. Ester. A density-based algorithm for discovering clusters in large spatial databases with noise: the uniqueness of a good optimum for k-means. In *SIGKDD*, pages 226–231, 1996.
- [5] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In *Advances in Knowledge Discovery and Data Mining*, pages 1–34. MIT Press, 1996.

<sup>6</sup>Deployed in the EGEE-III European Infrastructure Project (2008-2013) <http://www.grid-observatory.org/>.

[6] B. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.

[7] J. Gama, R. Rocha, and P. Medas. Accurate decision trees for mining highspeed data streams. In *SIGMOD*, pages 523–528, 2003.

[8] J. Gama and P. P. Rodrigues. Stream-based electricity load forecast. In *PKDD*, pages 446–453, 2007.

[9] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O’Callaghan. Clustering data streams: Theory and practice. *TKDE*, 15:515–528, 2003.

[10] Z. Harchaoui, F. Bach, and E. Moulines. Kernel change-point analysis. In *NIPS*, 2008.

[11] D. Hinkley. Inference about the change-point from cumulative sum tests. *Biometrika*, 58:509–523, 1971.

[12] D. Judd, P. K. McKinley, and A. K. Jain. Large-scale parallel data clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20:871–876, 1998.

[13] M. Leone, Sumedha, and M. Weigt. Clustering by soft-constraint affinity propagation: Applications to gene-expression data. *Bioinformatics*, 23:2708, 2007.

[14] M. Meila. The uniqueness of a good optimum for k-means. In *ICML*, pages 625–632, 2006.

[15] S. Muthukrishnan, E. v. d. Berg, and Y. Wu. Sequential change detection on data streams. In *ICDM Workshops*, 2007.

[16] E. Page. Continuous inspection schemes. *Biometrika*, 41:100–115, 1954.

[17] N. Palatin, A. Leizarowitz, A. Schuster, and R. Wolff. Mining for misconfigured machines in grid systems. In *SIGKDD*, pages 687–692, 2006.

[18] C. E. Rasmussen and C. K. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 01 2006.

[19] I. Rish, M. Brodie, and S. M. et al. Adaptive diagnosis in distributed systems. *IEEE Trans. on Neural Networks*, 16:1088–1109, 2005.

[20] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

[21] J. Villemonteix, E. Vazquez, M. Sidorkiewicz and E. Walter. Global optimization of expensive-to-evaluate functions: an empirical comparison of two sampling criteria. *Journal of Global Optimization*, vol 43 (2-3), p.373-389, 2009

[22] X. Zhang, C. Furtlehner, and M. Sebag. Data streaming with affinity propagation. In *ECML/PKDD*, pages 628–643, 2008.

[23] J. Andreeva, B. Gaidioz, J. Herrala, and et al. Dashboard for the LHC experiments. *Journal of Physics: Conference Series*, vol. 119, 2008.

[24] Real Time Monitor:  
<http://gridportal.hep.ph.ic.ac.uk/rtm/>.

[25] X. Zhang, C. Furtlehner, and M. Sebag. INRIA research report in progress.

## APPENDIX

### A. SCHEMATIC PROOF OF PROPOSITION 2.2

For the sake of readability and to lighten the argument, the influence between the center of mass and extreme value statistics distribution is neglected, enabling us to use a spherical kernel instead of cylindrical kernel and making no dis-

inction between  $ex$  and  $\tilde{e}x$ , to write the recurrence (see [25] for a complete discussion). Between level  $h$  and  $h + 1$ , one has:

$$f_{sd}^{(h+1)}(x) = \int_0^\infty K^{(h,M)}(x, y) f_{ex}^{(h,M)}(y) dy \quad (4)$$

with

$$\lim_{M \rightarrow \infty} M^{-1} K^{(h,M)}\left(\frac{x}{M}, \frac{y}{M}\right) = \frac{d}{\sigma^{(h)}} K\left(\frac{dx}{\sigma^{(h)}}, \frac{dy}{\sigma^{(h)}}\right) \quad (5)$$

where  $K(x, y)$  is the  $d$ -dimensional radial diffusion kernel,

$$K(x, y) \stackrel{\text{def}}{=} \frac{1}{2} x^{\frac{d-2}{4}} y^{\frac{2-d}{4}} I_{\frac{d-2}{2}}(\sqrt{xy}) e^{-\frac{x+y}{2}}.$$

with  $I_{\frac{d}{2}-1}$  the modified Bessel function of index  $d/2 - 1$ . The selection mechanism of the exemplar yields at level  $h$ ,

$$F_{ex}^{(h,M)}(x) = (F_{sd}^{(h)}(x))^M,$$

and with a by part integration, (4) rewrites as:

$$f_{sd}^{(h+1)}(x) = K^{(h,M)}(x, 0) + \int_0^\infty (F_{sd}^{(h)}(y))^M \frac{\partial K^{(h,M)}}{\partial y}(x, y) dy,$$

$$\text{with } \lim_{M \rightarrow \infty} M^{-1} K^{(h,M)}\left(\frac{x}{M}, 0\right) = \frac{d}{2\Gamma(\frac{d}{2})\sigma^{(h)}} \left(\frac{dx}{2\sigma^{(h)}}\right)^{\frac{d}{2}-1} \exp\left(-\frac{dx}{2\sigma^{(h)}}\right).$$

At this point the recursive hierarchical clustering is described as a closed form equation. Proposition 2.2 is then based on (5) and on the following scaling behaviors,

$$\lim_{M \rightarrow \infty} F_{ex}^{(h,M)}\left(\frac{x}{M^{\frac{d}{2}}}\right) = \exp(-\alpha^{(h)} x^{\frac{d}{2}}),$$

so that

$$\lim_{M \rightarrow \infty} F_{sd}^{(h+1)}\left(\frac{x}{M^\gamma}\right) = \lim_{M \rightarrow \infty} M^{1-\gamma} \int_0^\infty dy \int_{\frac{x}{\sigma^{(h)}}}^\infty du f_{\tilde{e}x}^{(h,M)}\left(\frac{y}{M^{\frac{d}{2}}}\right) K(M^{1-\gamma}u, \frac{M^{1-\frac{d}{2}}y}{\sigma^{(h)}}).$$

Basic asymptotic properties  $I_{d/2-1}$  yield with a proper choice of  $\gamma$ , the non degenerate limits of proposition 2.2. In the particular case  $d = 2$ , taking  $\gamma = 1$ , it comes:

$$\begin{aligned} \lim_{M \rightarrow \infty} F_{sd}^{(h+1)}\left(\frac{x}{M}\right) &= \int_0^\infty dy \int_{\frac{x}{\sigma^{(h)}}}^\infty du f_{\tilde{e}x}^{(h)}(\sigma^{(h)}y) K(u, y) \\ &= - \int_0^\infty dy \int_{\frac{x}{\sigma^{(h)}}}^\infty du \frac{de^{-\alpha^{(h)}\sigma^{(h)}x}}{dy} I_0(2\sqrt{uy}) e^{-(u+y)} \\ &= \exp\left(-\frac{\alpha^{(h)}}{1 + \alpha^{(h)}\sigma^{(h)}}x\right), \end{aligned}$$

with help of the identity

$$\int_0^\infty dx x^\nu e^{-\alpha x} I_{2\nu}(2\beta\sqrt{x}) = \frac{1}{\alpha} \left(\frac{\beta}{\alpha}\right)^{2\nu} e^{\frac{\beta}{\alpha}}.$$

Again in the particular case  $d = 2$ , by virtue of the exponential law one further has  $\alpha^{(h)} = 1/\sigma^{(h)}$ , finally yielding:

$$\beta^{(h+1)} = \frac{1}{2}\beta^{(h)}. \quad (6)$$