Sequential fault monitoring

Dawei Feng, Cecile Germain-Renaud, Julien Nauroy

Sequential fault monitoring

Dawei Feng
LRI
CNRS – Université Paris Sud
Dawei.Feng@lri.fr

Cécile Germain
LRI
and TAO, INRIA-Saclay
Cecile.Germain@lri.fr

Julien Nauroy
LRI
and TAO, INRIA-Saclay
Julien.Nauroy@lri.fr

***Abstract*—For large-scale distributed systems, the knowledge component at the core of the MAPE-K loop remains elusive. In the context of end-to-end probing, fault monitoring can be re-casted as an inference problem in the space-time domain. We propose and evaluate Sequential Matrix Factorization (SMF), a fully spatio-temporal method that exploits both the recent advances in matrix factorization for the spatial information and a new heuristics based on historical information. Adaptivity operates at two levels: algorithmically, as the exploration/exploitation tradeoff is controlled by a self-calibrating parameter; and at the policy level, as active learning is required for the most challenging cases of a real-world dataset.**

I. Introduction

We depend on computer systems that are not dependable: large scale distributed systems pervade real-world Information Technology infrastructures and usage; and, decades ago, Lamport characterized such systems as those where "*the failure of a computer you didn't even know existed can render your own computer unusable*".

Computer Science research has worked on large scale fault management since long, with two main directions: discovering faults, and coping with them. With the advent of truly massively distributed systems with complex structures, a key change now occurred: rich monitoring information becomes available. Complete knowledge, and the very concept, of the state of a distributed system remain unreachable for fundamental reasons [1]. But, with the availability of equally massive information, estimating elements of the system state becomes a realistic goal. Specifically, some fault management goals can be re-casted as *fault inference* problems. This work targets a specific aspect of fault monitoring, fault discovery. In the following, the inference approach to fault discovery will be called *fault prediction*, because it fits with the Collaborative Prediction [2] context, although fault estimation would more accurately describe the approach.

Predicting faults improves system availability and reliability by providing useful information for the next task of coping with them, as the systems are normally highly redundant and heavily supervised. Often, alternatives to the faulty services can be proposed [3], [4]; in these cases, a well organized fault management system will conceal the hardware and software dysfunctions and will provide a transparent service that is a crucial ingredient of Quality of Experience. On the other hand, irrecuperable faults must be signaled as fast as possible to the human of automatic supervision. Overall, this amounts to re-evaluate the role of monitoring in fault management, and to consider fault prediction as an *inference in the space-time domain*.

Autonomous Computing (AC) provides a conceptual framework for designing fault management for these monitoring-equipped systems. Its so-called MAPE-K loop is organized around a Knowledge component. To set up fault prediction as a realistic objective in an AC approach, the first question is which kind of knowledge is actually reachable. In a fault diagnosis approach, the knowledge component includes a detailed internal model of the system that can be exploited to pinpoint the faulty components. The root causes of the faults can be revealed through various techniques [5] like statistical inference [6], [7], [8], log-based causality analysis [9], [10], [11] or deterministic replay[12], [13], [14]. Fault diagnosis can be seen as the process of recognizing the most likely explanation for the symptoms based on some causal and effect models among the propositions of interest in the problem domain.

While diagnosis maximizes the usefulness of monitoring data, it faces some potentially significant practical limitations. The first one is simply scalability: diagnosis is NP-hard [6]. More profoundly, assuming knowledge of a decent model of the system internals might prove unrealistic (the "*computer you didn't even know existed*" nicely summarizes this feature). As a consequence, this work formulates the fault prediction problem in the context of end-to-end monitoring. The overall infrastructure is a black box, with no a priori knowledge of its structure. End-to-end probes are designed to test a functional property of this blackbox. Then, fault prediction involves a classification problem: from a selection of the probes (the training set), infer the outcomes of the other probes.

In many cases, the probes can be meaningfully replicated in the system. For instance, in the example that will be further described in section IV, the functionality is related to file access, and the probes are launched from the computing nodes to the storage nodes. Then, the replication takes a *matrix* form: the endpoints are the row- or column- entities, and the probes outcomes are the entries in this matrix. Formally, fault prediction becomes a *matrix completion* problem. The benefit with respect to a fully unstructured setting is that the matrix hypothesis grounds a Collaborative Filtering technique that is more powerful than behavioral clustering [15], [16]: although the exact causes of failures might remain elusive, causality can be precisely modeled as the rank of the matrix, to be inferred,

while clustering is bound to phenomenology and a-posteriori analysis of exemplars or centroids.

Our previous work [17], inspired by [18], addressed fault prediction in a classical Collaborative Prediction framework, as a purely spatial problem where the matrix is assumed to be a snapshot of the probes outcomes. Here, we take into account the fact that the system dynamically evolves at various time scales. Addressing this issue brings us closer to a model consistent with the practitioners expectations, but turns out to be significantly more difficult than the previous and more idealized setting. Not surprisingly, we cope with this difficulty by exploiting the dynamic setting for enriching the snapshot with time-related information: *sequential monitoring* deals with a sequence of partially observed matrices and makes prediction using information both from the current and previous time windows.

Within this framework, in order to be realistic, inference has to address two specific difficulties. Firstly, strongly imbalanced distributions must be assumed, as faults are hopefully much less represented that nominal behavior; this belongs to the spatial aspect of inference. Second, in the time domain, one cannot assume that measurements could be kept fully up-to-date, as these systems are highly dynamic environments.

Fortunately, the same adaptivity strategy proved successful in various contexts to address both imbalanced distributions and noisy information: active learning iteratively selects most-informative samples in order to best improve the prediction accuracy. On the other hand, and always with realism in mind, active learning has the drawbacks to slow down the fault discovery process, as it requires to build the input incrementally, and to make it more complicated, thus more fault-prone itself. A transversal goal of this work is thus to evaluate the specific contribution of the active learning ingredient in the fault inference methods that we propose.

Our experimental validation dataset comes form the European Grid Initiative (EGI). Grids tend to be regarded as somehow outdated, thus a few words about the relevance of the dataset might be necessary. The specific grid technologies of the 2000's have of course been superseded by cloud-related ones. However, the essential paradigm of grid is organized sharing: safely and fairly federating hardware, software and data resources from multiple independent providers. Thus grids exemplify both the physical problems of worldwide scale systems, and the major issues of a multi-owned multi-operated system, that are equally present in federated clouds.

The main contributions of this paper is the **SMF** (Sequential Matrix Factorization) algorithm, and its active learning version, **SMFA** (Sequential Matrix Factorization with active learning), that efficiently combines the spatial and temporal information sources. Its major strength is to balance exploration and exploitation with a self-calibrating parameter that formalizes and exploits the multi-scale intuition of the practitioners. The adaptive balancing does not suffer form the same drawbacks as active learning, as the required information does not need on-line querying.

The paper is organized as follows. Section II presents the matrix completion context and the empirical motivation for going beyond a pure matrix completion approach. Section III describes SMF and its active learning version, SMFA. The next three sections present a detailed experimental evaluation, where the vanilla algorithms are combined in various ways with agnostic optimizations (smoothing) and information-oriented ones (strategies for active learning). Section IV describes the experimental setting, and sections V and VI the results on the EGI dataset, before the usual conclusion.

II. Contexts and motivations

A. Matrix factorization for fault prediction

Consider the simple setting where a partial snapshot of the system is available through end-to-end probes (e.g. a *ping*). Assume that we have M sources and W targets. A *probe selection* method defines which ones of the possible MW probes are actually launched. Each individual result is binary: *positive* means that the probe *failed*, i.e. a fault occurred; *negative*, the probe succeeded. Let X be the sparse $M \times W$ binary matrix of the outcomes of the selected probes.

Then, the inference task falls into the general category of *matrix completion*: given a sparse matrix X, find a full matrix Y of the same size that approximates X, i.e. that minimizes some measure of discrepancy between X and Y. When Y is required to be equal to X on the known entries, the problem is termed *exact completion*, and *approximate* otherwise.

With such a general setting, the problem is hopelessly ill-defined: in order to guess the missing entries, some assumptions have to be made about the matrix to recover Y. A natural one is to look for *low-rank* matrices, amounting to assume that a small number of hidden and partially shared factors (*latent* factors) affect the matrix entries.

The existence and unicity of a solution of exact matrix completion is a complicated problem (see e.g. [19]). Anyway, this formulation does not look very helpful, as rank minimization for completion is NP-hard and not feasible practically even for small sizes. However, it has paved the way for efficient algorithms, both for exact [19], [20] and approximate [21] completion. The main insight is to replace the rank by the trace (or nuclear) norm in the regularization term. For approximate completion, with hinge loss as the discrepancy measure, this yields the **Maximum Margin Matrix Factorization (MMMF)** algorithm [21] with the objective function:

$$||Y||_\Sigma + C\mathcal{L}_h(X(S), Y(S)), \tag{1}$$

where S is the set of known entries, $||.||_\Sigma$ is the trace norm, and $\mathcal{L}_h(A, B) = \sum_{ij \in S} \max(0, 1 - A_{ij}B_{ij})$ is the hinge loss between A and B. This formulation, unlike the low-rank approximation, is convex, thus is guaranteed to find the global optimal solution. More generally, as the trace norm is a convex function, and the matrices of bounded trace norm are a convex set, any convex loss function provides a convex optimization problem. However, each observed entry acts as a constraint during the minimization process, and the computational complexity increases drastically with the

number of involved constraints. Therefore, the external cost of each probe as a monitoring overhead doubles as an internal computational practical limit.

In the fault inference case, as the minimization procedure of Eq. (1) produces a real-valued matrix, a decision threshold (the classification common practice) gives the binary matrix of predicted behavior that represents the inferred outcomes of launching all MW probes.

In classical collaborative filtering, X is given (e.g. consumer ratings). With fault inference, there are more degrees of freedom: the goal is to optimize the tradeoff between the number of actually launched probes and the prediction quality.

B. Motivation

[18] proposed a method to handle fault inference based on a collaborative prediction approach, further analyzed in [17]. Although this method significantly reduces the number of required probes for acquiring an accurate view of the system, it is somehow static. Specifically, its only input is a snapshot from (assumed) simultaneous probes.

This setting has two drawbacks. Firstly, fault behavior is multi scale in time: beyond the stable system components with consistent status over time, there are other ones which status may fluctuate intensely at peak time and remain stable at off-peak time. Second, transient faults are systematically observed: transients are faults that get on and off at high frequency and should be considered as noise; practitioners do not have a clear explanation for them, and they might as well be produced by flaws in the monitoring software itself. Of course, the problem is to disentangle them from real, but short-lived faults.

A further motivation is to explore the possibility of getting rid of adaptivity. [18] integrated active learning with MMMF (min-margin heuristic) and [17] showed that active learning was a required ingredient in the most difficult and realistic case on a real-world fault prediction example. On the other hand, active learning is somehow inconvenient: because the probe selection is adaptive, it requires a feedback loop, an interface between online analysis and monitoring, resulting in a more complicated software than with a pre-determined setting. At grid or cloud scale, any unnecessary source of complexity should be eliminated. Thus, we explore the hypothesis that the past could statically provide information equivalent to the one obtained by adaptively querying the present.

To summarize, we explore two questions: 1) can we improve fault inference by integrating historical information within the matrix completion framework; and 2) does the active leaning component remain useful in this case?

III. Sequential matrix predictionn

A. Problem statement

There are two types of information available for sequential monitoring: spatial and temporal information. The spatial information can be thoroughly exploited by a collaborative prediction method like MMMF, while on the other hand, the temporal information which concerns the dynamics of the

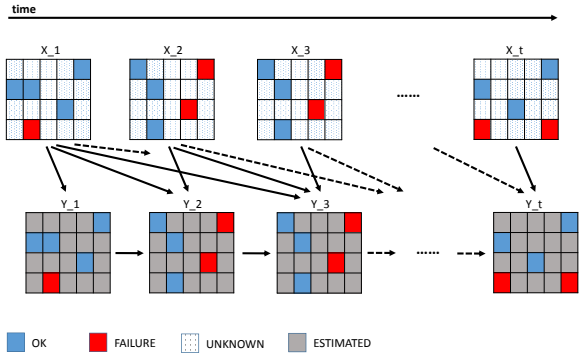


Fig. 1. Overview: at each time step, the current probes provide known results; simple matrix completion would use only these, while a spatio-temporal method exploits the history of probes (are they consistent?) and the history of inferences (were they accurate?).

entries provides extra opportunity for improving algorithm performance. Let

$X_t \in B^{M\times W}$ be the partially observed matrix at time t,
$\hat{Y}_t \in R^{M\times W}$ be the result of a prediction algorithm,
$Y_t \in B^{M\times W}$ be the thresholded binary version of $\hat{Y}_t$.

With threshold ρ, $Y_T(i,j)$ is defined by:

$$Y_T(i,j) = \begin{cases} -1 \;\; if \quad \hat{Y}_T(i,j) \leq \rho, \\ 1 \;\; \text{otherwise.} \end{cases}$$

With $B = \{-1, 1\}$, the binarization threshold ρ is set to 0.

We define the task of *sequential matrix prediction* as: given a series of partially observed matrices $(X_1, \ldots, X_t)$, predict the fully estimated matrix Y_t. Figure 1 illustrates the sequential process: at each time step t, a matrix Y_t is estimated from the observation sequence $(X_1, \ldots, X_t)$ and the sequence of the past estimates $(Y_1, \ldots, Y_{t-1})$.

B. The SMF algorithm

At each time step t we have a sequence of history predictions $Y_1, ..., Y_{t-1}$. The confidence in these predictions can be expressed by the distance of each predicted value to the separation hyper-plane. Thus two types of predictions emerge: those predictions close to the separation plane and those far from the separation plane. We call the former ones the *most uncertain* prediction set and the latter ones the *most confident* prediction set. From the system point of view, the most uncertain predictions are related to those components with short term status like the transient faults and the most confident predictions are related to those components with relatively long term stable status.

In this section, we propose an algorithm, i.e. sequential matrix factorization (SMF), to capture both the long term and short term status behavior by utilizing the spatial information

as in MMMF, and exploring the most uncertain and most confident heuristic concealed in the temporal information meanwhile. In the following, S_u, S_c and S_r are index sets of matrix X, for denoting the most uncertain prediction set, most confident prediction set and a random sample set, respectively. The observed set (labels queried at time t) is $S_u \cup S_r$. All three sets depend on t, but we dropped the unnecessary supplementary indices.

Recall that the objective function of MMMF is composed of two terms (Eq. 1): the first one is the trace norm of the estimated matrix Y_t and the second term is the hinge loss between estimation and observation. In the following we will develop the objective function of SMF by adding the *most uncertain* and the *most confident* information to Eq. 1.

First we consider the most uncertain information. The most uncertain prediction set S_u (entries with small margin to the classification hyper-plane) can be derived from Y_{t-1} and their labels at time t can be queried from the system. Hence, the ground truth of those most uncertain predictions in Y_{t-1} is available in the sample set X_t. We denote this as $X_t(S_u)$.

The second information, i.e., the *most confident* predictions, is concealed in the history estimation. For these most confident entries, instead of sampling their true labels at time t, their previous predictions can be used directly in the next run. Specifically, in SMF we choose those most confident predictions from Y_{t-1} and assume their states remain unchanged at time t with a confidence level γ.

We compute γ in terms of the overall difference between Y_{t-1} and X_t (i.e. the difference between last estimation and current observation) on the *observed* entries in X_t. Any classification criteria like accuracy, true positive rate (TPR) or FSCORE can be used for measuring this discrepancy. Since in X_t the observed set is $S_u \cup S_r$, we therefore compute γ as the difference between $X_t(S_u \cup S_r)$ and $Y_{t-1}(S_u \cup S_r)$ as follows:

$$\gamma = \text{TPR}(Y_{t-1}(S_u \cup S_r), X_t(S_u \cup S_r)), \tag{2}$$

where $\text{TPR}(A, B)$ is the true positive rate of A according to the ground truth set B. In the prediction, γ is used as an adaptive cost ratio which adjusts the weight (penalty) of the heuristic information in the objective function (similar to the coefficient C in Eq. 1). The reason we choose *TPR* as the penalty lies in the fact that in distributed system monitoring successfully discovering a failure comes more important than alerting one incorrectly (see section IV-C).

In addition to the most uncertain set S_u and most confident set S_c, we keep the random set S_r in the objective function of SMF. The random sample set S_r serves as a term for avoiding over-fitting the history information, as sudden change between past estimation and current observation might occur. To sum up, SMF has the following objective function:

$$||Y_t||_\Sigma + C\mathcal{L}_h(Y_t(S), X_t(S)) + C\gamma\mathcal{L}_h(Y_t(S_c), Y_{t-1}(S_c)), \tag{3}$$

where $S = S_u \cup S_r$ is the sample set which labels are queried at time t and S_c is the most confident prediction set that we borrow from $t-1$. Thus the difference between Eq. 1 and 3 is exhibited by the selection of S_u and the presence of S_c. As in Eq. 1 this function is convex, thus can be directly minimized.

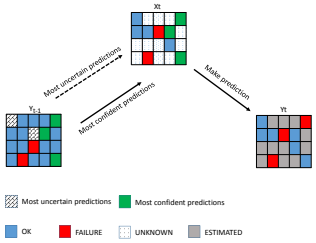


Fig. 2. SMF heuristic: the most confident predictions are directly used, the most uncertain are probed.

Figure 2 illustrates the selection process. The most uncertain and most confident predictions are selected from Y_{t-1}, where labels of the former set are further queried at time t, and labels of the latter set are the estimation values in the last run.

Algorithm 1 describes the pseudocode of SMF. At the beginning, the sample set S of X_t is generated by a combination of selecting most uncertain predictions from Y_{t-1} and a random sampling (line 1 to 3). Then the true labels of S are queried from the system and are used as ground truth for measuring the discrepancy between Y_{t-1} and X_t (line 4, 5). The most confident predictions in Y_{t-1} are selected in the following step and used as input for the estimation. In the final step Y_t is derived by finding an estimation which minimizes Eq. 3.

C. Sequential matrix factorization with active sampling, SMFA

In active matrix factorization [18], [17], the prediction performance is improved by selecting the sample entries in X_t actively and iteratively, using the most uncertain heuristic from the very last prediction until the maximum allowed number of samples is reached. The key idea is that, with the progress of each iteration, confidence in the estimation increases simultaneously.

In SMF, sample entries are selected under three policies: random, most uncertain and most confident. The latter two strategies rely on information from the last prediction Y_{t-1}. The selection of active samples is complete all at once in SMF and no further actions can be taken given its first estimation of Y. **Sequential matrix factorization with active sampling (SMFA)** builds a sequence of estimators $Y_t^i, i = 2, 3, ...$ that iteratively benefits form the estimation process to refine the definition of both the most uncertain and most confident predictions.

Algorithm 2 describes the steps of SMFA. We denote the estimation matrix at the ith iteration of time t as Y_t^i. At the beginning, SMF is used to give an initial estimation Y_t^0 from Y_{t-1} (line 4), then an iterative estimation is employed on the prediction sequence $Y_t^i, i = 1, 2, ...$ until the maximum number of samples is reached (line 5 to 9). Active sample

Algorithm 1: SMF, Sequential Matrix Factorization

Input: $\mathbf{Y_{t-1}}$, last prediction;
$\mathbf{N_u}$, number of most uncertain samples from Y_{t-1};
$\mathbf{N_c}$, number of most confident samples from Y_{t-1};
$\mathbf{N_r}$, number of random samples;
$\mathbf{C}$, slack penalty.
Output: Full real-valued matrix $\mathbf{Y_t}$
Initialize: $Init\ h_1, h_2, h_3$, /*Initialize the most uncertain, most confident and random sampling heuristic, respectively*/;

$S_u \leftarrow Sample(h_1, N_u, Y_{t-1})$ /*select N_u most uncertain sample indexes from Y_{t-1}*/;
$S_r \leftarrow Sample(h_2, N_r)$, /*select N_r random sample indexes*/;
$S \leftarrow S_u \cup S_r$;
$X_t(S) \leftarrow QueryLabels(S)$, /*query the true label for entries in S*/ ;
$\gamma \leftarrow TPR(X_t(S), Y_{t-1}(S))$ /*given $X_t(S)$ (true labels for entries in S), compute the true positive rate of $Y_{t-1}(S)$*/;
$S_c \leftarrow Sample(h_3, N_c, Y_{t-1})$, /*select N_c most confident samples from Y_{t-1}*/;
$Y_t \leftarrow \arg\min_Y ||Y_t||_\Sigma + C\mathcal{L}_h(Y_t(S), X_t(S)) + C\gamma\mathcal{L}_h(Y_t(S_c), Y_{t-1}(S_c))$ /*find an estimation that minimizes the objective function*/;
return Y_t

Algorithm 2: SMFA, Sequential Matrix Factorization with active sampling

Input: N, max # of new samples;
Y_{t-1}, last prediction;
P_0, initial sample rate for the 1st prediction;
P_a, active sample rate at each iteration;
ρ, ratio of random samples and most uncertain samples for P_a;
C, slack penalty.
Output: Full real-valued matrix Y_t
initialize: $Init(N_c)$. /*Initialize the number of most confident samples to select in each iteration*/;

$i = 0$ /*current iteration index*/ ;
$n = N \times P_0$ /*current number of new samples*/ ;
$[N_u, N_r] \leftarrow getSampleSize(n, \rho)$ /*Get random and most uncertain sample size for the initial prediction*/;
$Y_t^i \leftarrow SMF(Y_{t-1}, N_u, N_c, N_r, C)$;
while $(n < N)$ **do**
  $[N_u, N_r] \leftarrow getSampleSize(N \times P_a, \rho)$ /*Get random and most uncertain sample size according to ρ and P_a*/;
  $Y_t^{i+1} \leftarrow SMF(Y_{t_i}, N_u, N_c, N_r, C)$;
  $n = n + N_u + N_c + N_r$;
  $i = i + 1$;
$Y_t = Y_t^i$;
return Y_t

selection is engaged each time the SMF algorithm selects the most uncertain and most confident predictions from the last estimation.

D. Smoothing the results

Although one of the key features in SMF or SMFA is to preserve the continuity of predictions between consecutive time windows, extra smoothing of the outputs can be considered. Smoothing the prediction sequence with exponentially weighted moving average (EWM) works as follows:

$$Y'_k(i,j) = \begin{cases} Y_k(i,j), k = t-l+1, \\ \theta Y_k(i,j) + (1-\theta)Y'_{k-1}(i,j), k = t-l+2, ..., t \end{cases}, \quad (4)$$

where $\theta \in (0,1)$ is a user-defined damping factor, and l is the lag window length.

E. Methods summary

Table I summarizes the methods introduced in the two previous sections, with their inputs, outputs and related parameters. For a given method H, its smoothed version is noted H^* (e.g. the smoothed version of SMF is noted SMF* in later section).

TABLE I
METHODS SUMMARY

	Input Data	Output Data	Parameters
MMMF	X_t	Y_t	N, # of samples; C, slack penalty;
SMF	X_t, Y_{t-1}	Y_t	N_r, # of random samples; N_c, # of most confident samples. N_u, # of most uncertain samples; C, slack penalty.
MMMFA	X_t	Y_t	N, # of total samples; C, slack penalty; P_0, initial sample rate; P_a, active sample rate at each iteration; ρ, ratio of random sample and most uncertain sample for P_a;
SMFA	X_t, Y_{t-1}	Y_t	N, # of total samples; C, slack penalty; P_0, initial sample rate; P_a, active sample rate at each iteration; ρ, ratio of random sample and most uncertain sample for P_a;
H^*	$Y_{t-L+1}, ..., Y_t$	Y'_t	L, lag window length; θ, damping factor for smoothing.

IV. Experimental setting

A. The source

The European Grid Infrastructure (EGI) enables access to computing resources for European researchers from all fields of science, including high energy physics, humanities, biology and more. The infrastructure federates some 350 sites worldwide, gathering more than 250,000 cores, which makes it the largest non-profit distributed system worldwide. Hardware and software failures are intrinsic to such large-scale systems. Middleware e.g. gLite [22], Globus [23] or ARC [24] cannot handle this without substantial human intervention. Access rights to EGI are primarily organized along the concept of Virtual Organization (VO), and each of the 200 VOs has to be specifically configured on its supporting sites, which adds complexity and introduces extra failures. User communities exploit two strategies to cope with faults: overlay middleware e.g. DIRAC [25], DIANE [26], AliEn [27] and PaNDA [28] implements specific fault-tolerance strategies to isolate users

from the vagaries of the infrastructure; and monitoring identifies problems and quantifies performance w.r.t. quality of service agreements.

The data source for this study is the Biomed VO. Biomed has access to 256 Computing Elements (CEs) and 121 Storage Elements (SEs). CEs are shares of computing resources, implemented as queues of each site manager (e.g. PBS), and SEs are shares of storage resources; the formal definition is part of the Glue Information model [29]. File access remains one the major issues, for numerous causes spanning from hardware breakdowns to entanglements in the complicated verification of access rights, and including the bizarre transients reported by operations managers.

B. Data description

The dataset[1] was collected on EGI by submitting a series of jobs to 212 Biomed CEs every two hours between *Mon Nov 12 15:52 CET 2012* and *Sat Nov 24 09:54 CET 2012*, for about 282 hours in total. Each job tested the service availability between its CE and each of 96 Biomed SEs, by launching the *lcg-cp* probe. lcg-cp copies a file (like Unix cp), thus is a relatively high-level probe that tests core access (network path), availability of the access control services as well as reading and writing capacities. The CEs and SEs have been preselected as relatively reliable, in order to eliminate trivially discoverable faults.

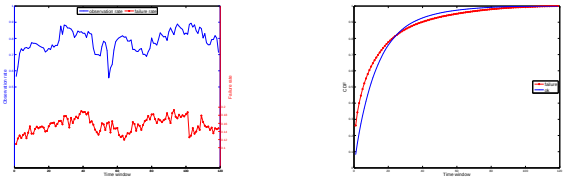
Of course, our test jobs were fully protected again the consequences of a *probe* failure (the job successful termination does not depend on the outcome of the probe), and the procedure has been designed so that the resources involved in running the jobs are as disjoint as possible from those required by the probes. However, our test jobs were no more immune to *middleware* faults than any other user job, and a significant part of them did fail, thus reporting no information at all. In order to get a consistent sample, we deleted the data from those CEs with less than 7000 observed entries and also from those time windows with less than 50% data observed. This results in a data cube of size $79 \times 96 \times 119$, with each dimension corresponding to CE, SE, and time window respectively. The goal of our experiment is to predict whether the jth SE is accessible from the ith CE at a given time window t, and this data cube is the *ground truth* for the prediction algorithms. We use 0 for representing a missing observation, 1 for a *Failed* probe (job succeeded and *lcg-cp* failed), and -1 for an OK probe. This notation is in accordance with the general meaning of positive (abnormal) and negative (normal) in statistics.

Let M be the total number of CEs, W be the total number of SEs, and $t_{k,k=1,2,\ldots,T}$ be the time window sequence, we further note N_{t_k} as the number of observed entries at t_k and $N^+_{t_k}$ be the number of positive entries (failures) at t_k, then the observation rate and test failure rate at t_k are defined as $r_{t_k} = N_{t_k}/MW$ and $f_{t_k} = N^+_{t_k}/MW$, respectively. Figure 3(a) illustrates the *observation rate* r_{t_k} and *failure rate* f_{t_k} of the dataset. Most of the observation rates stay above 70% and the failure rates are less than 20%. A high observation rate ensures a more reliable result for performance evaluation, since we have more ground truth information at hand. A relatively stable failure rate indicates a consistent system status in consecutive time windows. The failure rate presents some breakpoints, e.g. the sharp drop from 18.74% to 12.57% at the 101st time window. Their impact on prediction performance is discussed in sections V and VI.

[1]The dataset will be publicly available of the Grid Observatory web site (www.grid-observatory.org) from April 2014.

TABLE II
ILLUSTRATION OF DURATION LENGTH FOR OK AND FAILURE

sequence	1	1	0	1	1	1	-1	0	-1	-1
	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
duration	2	1		3	2	1	1		2	1



(a) Observation rate and failure rate over time

(b) CDF of duration of OK and Failure

Fig. 3. Empirical statistics of the dataset.

Another interesting aspect of the data is the *duration length* of each status (i.e. OK or Failed). The duration length of a status is defined as the number of time windows the status spans until a different status is observed in the sequence; Table II gives an example, and Figure 3(b) shows the cumulative distributions. The high proportion of duration length 1 for Failed (about 25%) illustrates the phenomenon of transients, while the about 20% entries for both success and failure with a duration length at least 26 illustrate relative stable behavior. These distributions sustain our considerations about a multi-scale in time behavior.

C. Criteria

With the availability of the ground truth, the classical performance indicators for binary classification can be measured. Accuracy (the ratio of correctly predicted entries over the total number of entries) is of limited interest: as the data set is not balanced, overly optimistic algorithms that favor OK predictions will exhibit satisfactory accuracy. The indicators associated with the risks (confusion matrix) are more informative: *sensitivity* (True Positive Rate), the proportion of actual positives that are correctly predicted; *specificity*, the proportion of actual negatives that are correctly predicted; *precision*, the ratio of true positives over all predicted positives.

They all make sense for operational needs, but sensitivity is the most important one: an undetected failure (bad sensitivity) results in a failed user job and a dissatisfied user, while a false negative might go unnoticed; specifically when the services are

TABLE III
SUMMARY OF PARAMETER VALUES

	Parameters
MMMF	$N = 10\%$ of random samples in X_t; $C^+ = 10$, coefficient for slack penalty.
SMF	$N_r = 5\%$ of random samples in X_t; $N_u = 5\%$ of most uncertain samples in X_t; $N_c = 10\%$ of most confident samples from Y_{t-1}; $C^+ = 10$, slack penalty.
MMMFA	$P_0 = 5\%$, initial sample rate; $P_a = 1\%$, active sample rate at each iteration; $\rho^+ = 0.5$, equal size of random samples and most uncertain samples at each active iteration; $C^+ = 10$, slack penalty.
SMFA	$P_0 = 5\%$, initial sample rate; $P_a = 1\%$, active sample rate at each iteration; N_c, 10% of most confident prediction from Y_{t_i}; $\rho^+ = 0.5$, equal size of random samples and most uncertain samples at each active iteration; $C^+ = 10$, slack penalty.

redundant (e.g. from replicated data), the failed request can be transparently rerouted to another server. Of course, specificity is nonetheless an important criterion, and compound criteria such as precision, or Fscore measure various tradeoffs between them.

The last important indicator is the Matthews Correlation Coefficient (MCC), a correlation coefficient between the observed and predicted binary classifications that is relatively insensitive to unbalanced classes. Its interest comes from the fact that MCC is a proxy for the Area Under ROC (Receiver Operating Characteristic) Curve (AUC), which summarizes the intrinsic quality of a binary classifier independent of the decision threshold. Moreover, MCC does not assume that the classification error is a reasonable estimation of the prediction error [30], thus is a more robust indicator with respect to the objective functions involved in the optimization step of the algorithms. Other related indicators such as those involved in Neymann-Pearson learning [31] have not been reported, as they have not still gained widespread acceptance.

D. Details

All experiments are performed 10 times and results are reported their average. For all algorithms, 10% of the total entries are selected as training set (i.e. $N = 10\% \times W \times M$), and for SMF and SMFA another 10% of the most confident entries with values from Y_{t-1} is added as in algorithm-1. This supplementary information does not require any probe: it only exploits the past (inferred) outcomes. The adaptive weight for the most confident entries is computed according to line 5 in algorithm-1. Table III lists the concrete parameter settings for the algorithms of Table I. Parameters marked with a '+' are selected via training and validation on the first 20 time windows.

V. EXPERIMENTAL RESULTS - NON-CURATED DATASET

We first analyze the vanilla algorithms, then evaluate the impact of active learning and of smoothing. Some supplementary material is available in [32].

A. Vanilla methods

The results of Table IV can be analyzed along different paths. Firstly, while both algorithms reach fairly good specificity, sensitivity exhibits only acceptable performance: 25-30% of the actual failures would not be predicted. This is a natural, but nonetheless problematic effect of the imbalanced dataset.

TABLE IV
PERFORMANCE COMPARISON FOR VANILLA ALGORITHMS

	Sensitivity	Specificity	Precision	MCC	FSCORE
MMMF	0.713±0.040	0.970±0.010	0.824±0.045	0.725±0.051	0.764±0.041
SMF	**0.747±0.047**	**0.985±0.006**	**0.901±0.038**	**0.791±0.046**	**0.816±0.040**

SMF shows a significant advantage on sensitivity, which is our primary performance indicator (c.f. section IV-C) and is also better for specificity, translating into an altogether clear advantage on the compound indicators.

The time series presented in Figures 4(a) and 4(b) provide some insight on the factors of performance. MMMF and SMF behave essentially in lockstep on sensitivity, showing that matrix factorization provides a decisive contribution; the most important gains of SMF over MMMF occur in the relatively stable intervals (e.g. 40-55) where the knowledge of the past matters. This is also true for specificity, except at the sharp drop of SMF at the 81st time window (recall that we use 20 time windows as initial input, so this is the 101st time window in the original data). As mentioned in IV-B, this is caused by a drop in real failures between the observations in the two adjacent time windows. In this case the historical information does not help, but instead hinders performance improvement, biasing the algorithm towards false positives.

B. Active methods

TABLE V
PERFORMANCE COMPARISON WITH ACTIVE SAMPLING

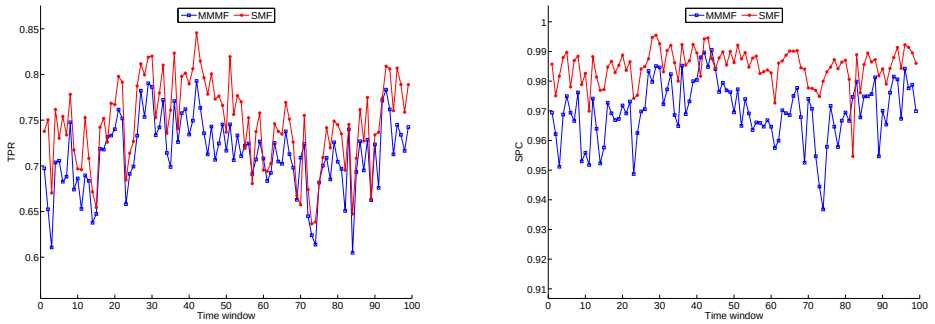
	Sensitivity	Specificity	Precision	MCC	FSCORE
MMMF	0.713±0.040	0.970±0.010	0.824±0.045	0.725±0.051	0.764±0.041
MMMFA	0.789±0.037	0.959±0.013	0.800±0.048	0.752±0.052	0.793±0.041
SMF	0.747±0.047	**0.985±0.006**	0.901±0.038	0.791±0.046	0.816±0.040
SMFA	**0.826±0.047**	0.983±0.007	**0.907±0.033**	**0.840±0.046**	**0.864±0.038**

As explained in section III-C, active learning is a candidate for improving on sensitivity. Table V compares MMMF and SMF with their active versions. In both cases, sensitivity improves by 11%, while the decrease in specificity is negligible (respectively 1% and 0.2%). Moreover, SMFA outperforms MMMFA, but active learning is powerful enough to make MMMFA outperform SMF on sensitivity by 6%. In other words, the good selection of current information allows to forget the past: for selecting the most uncertain prediction, which is likely to be on the positive entries, active sampling on the current data does a better job than passive history.

C. Smoothing

TABLE VI
PERFORMANCE COMPARISON WITH SMOOTHING

	Sensitivity	Specificity	Precision	MCC	FSCORE
MMMF*	0.700±0.045	0.990±0.004	0.933±0.031	0.778±0.041	0.799±0.037
MMMFA	0.789±0.037	0.959±0.013	0.800±0.048	0.752±0.052	0.793±0.041
MMMFA*	0.788±0.039	0.987±0.005	0.924±0.029	0.826±0.038	0.850±0.032
SMF*	0.716±0.071	**0.993±0.005**	0.947±0.042	0.797±0.053	0.813±0.051
SMFA	0.826±0.047	0.983±0.007	0.907±0.033	0.840±0.046	0.864±0.038
SMFA*	**0.827±0.047**	0.991±0.005	**0.950±0.028**	**0.865±0.041**	**0.884±0.036**



(a) Sensitivity (b) Specificity

Fig. 4. Time series of sensitivity and specificity for the vanilla algorithms. Windows are numbered from the first one where prediction starts.

As explained before, one can wonder whether if simple smoothing would not be competitive with the active approach, with much less complexity. In fact, smoothing actually often degrades sensitivity even with respect to the vanilla algorithm (e.g. MMMF goes from 0.713 down to 0.700): the smoothing process over-corrects the false positive predictions.

Table VI compares the active versus smoothing approach (e.g. SMFA vs SMF*); clearly smoothing is not competitive with active learning on sensitivity, although it always improves specificity and sometimes the compound criteria. On the other hand, combining smoothing and active learning has contrasted results, degrading MMMFA sensitivity, but marginally improving on SMFA.

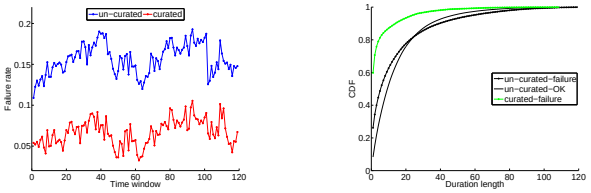
D. Summary

The above analysis firstly emphasizes the effectiveness of the sequential matrix factorization approach, and specifically of the proposed SMF algorithm and its variants: a properly synthesized use of spatial and temporal information significantly outperforms a purely spatial method. Then, active learning is consistently and significantly beneficial. As the positive entries are the minority part of the whole population, it is therefore difficult to uncover them by using any conventional method with equal cost on positive and negative entries. However, with the aid of active sampling it is possible to unveil those *difficult to predict* entries, since they are more likely to be exposed and labeled during the active sampling process. Finally, simple smoothing cannot compete with the active approach, and should be considered useful only combined with active learning, and only when false positives are a major concern.

VI. Experimental results with the curated dataset

A. The curated dataset

It could be argued that our benchmark is too easy: the tail of the distribution of the failure duration lengths corresponds to long-lasting errors, that basic monitoring tools (e.g. heartbeats) would report anyway, and the prediction methods should be applied only to more elusive causes of errors. While this is disputable (remember that all probes succeed as jobs, thus a significant part of the services are up and running), it is worth assessing the performance of the methods when these systematic errors are eliminated. Therefore, we designed a second set of experiments, with *curated* matrices as the reference fault structure.



(a) Timeseries of failure rates (b) CDF of duration lengths

Fig. 5. Comparison of curated and un-curated datasets.

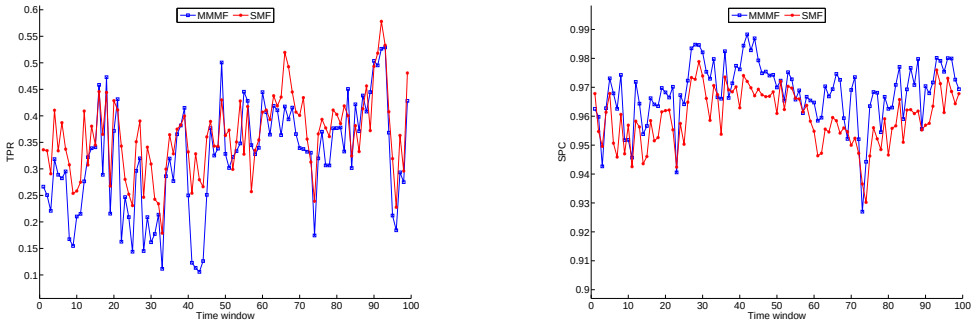
TABLE VII
Performance comparison on the curated dataset

	Sensitivity	Specificity	Precision	MCC	FSCORE
MMMF	0.319±0.102	0.968±0.011	0.427±0.110	0.328±0.106	0.361±0.107
MMMFA	0.482±0.081	0.959±0.014	0.471±0.080	0.436±0.080	0.471±0.077
SMF	0.362±0.074	0.960±0.009	0.374±0.078	0.326±0.075	0.365±0.074
SMFA	**0.569±0.076**	**0.986±0.006**	**0.743±0.079**	**0.628±0.080**	**0.642±0.076**

The curated dataset is derived by removing those lines and columns with at least 98% failed entries in the reference matrices. Figure 5(a) shows the magnitude of the decrease in the failure rate, approximately from 15% to 5% on average. Moreover, the CDF of failure duration length also experiences a sharp change (Figure 5(b)). The percentage of length-one durations increases from 25% to about 60%, and the percentage of duration lengths less than 20 grows from 75% to approximately 92%. In other words, after the elimination of systematic failures, the proportion of short term failures increases significantly.

B. Vanilla and active methods

The very adverse curated dataset produces quite interesting results (Table VII) concerning sensitivity. Most importantly, all vanilla method perform poorly: more than 60% of the



(a) Sensitivity (b) Specificity

Fig. 6. Curated dataset: time series of sensitivity and specificity for the vanilla algorithms. Windows are numbered from the first one where prediction starts.

TABLE VIII
PERFORMANCE AT DIFFERENT SAMPLING RATE, CURATED DATASET

	Sensitivity	Specificity	Precision	MCC	FSCORE
SMFA*-0.1	0.562±0.078	0.993±0.003	0.853±0.053	0.675±0.069	0.675±0.070
SMFA*-0.15	0.664±0.076	0.997±0.002	0.947±0.031	0.780±0.057	0.778±0.059
SMFA*-0.2	**0.720±0.074**	**0.998±0.002**	**0.970±0.033**	**0.825±0.051**	**0.825±0.054**

faults will get unpredicted. The extreme imbalance of the data obviously highlights the limits of accuracy-based methods (MMMF).

Figure 6 give some insights about the issues encountered by SMF. For sensitivity, contrary to the non-curated case, SMF and MMMF are not in lockstep. More precisely, there are intervals where they go in opposite directions, e.g. at time 40-45; there, taking into account the past somehow helps SMF to limit its loss, but the impact of the catastrophic behavior of MMMF is still too strong.

As can be expected, active learning procures a decisive improvement, in the order of more than 50% for both. SMFA has a clear advantage. It capable of discovering more than 50% of the faults while maintaining good specificity and acceptable precision: only 25% of the alarms are spurious.

C. Higher sampling rate

So far, the sampling rate was limited to 10%. Table VIII illustrates the result of increased sample rates, i.e. 10%, 15% and 20%, of SMFA* on the curated dataset. Results are averaged on a 5-run experiment. A steady and notable improvement is exhibited. For example, at 20%, SMFA* finds out 72% of the faults while keeping a balanced MCC value of 0.825. The good news is that the prediction performance increases steadily with the sample rate even in the curated situation. However, the sampling rate drives two costs: the monitoring overhead per se, and the computational complexity, as discussed in section II-A. To control the computational cost, accelerating methods like Fast MMMF [33] might be required.

D. Summary

The curated dataset is highly imbalanced, with only about 5% positive entries. Then static accuracy-based classification becomes awkward and exhibits poor performance. Active learning is one of the few approaches that can contribute to alleviate this issue. Our question in this section was its impact within already "smart" - heuristic based - methods. The results show that active learning carries a consistent improvement, with a very similar factor, for the two relatively different heuristics involved with SMF and MMMF; in other words, active learning is robust in this context.

VII. CONCLUSION

Efficient monitoring of production grids and clouds at acceptable manpower cost cannot assume exhaustive a priori knowledge of their software and hardware infrastructures. In this context, and with end-to-end probing as sole data acquisition strategy, fault discovery can be re-casted as an inference task, and can borrow methods from collaborative prediction. The challenge as well as the opportunity brought by switching from a static, snapshot-oriented, view of the monitoring to considering the time dimension reside in the sequential correlation between consecutive data points. We have shown that these sequential patterns, if exploited properly, can play an important role in improving prediction performance.

This paper explores various methods that combine time and space information with increasing complexity. It proposes and evaluates SMF, a fully integrated method that exploits both the recent advances in matrix factorization for the spatial information and a new heuristics based on historical information. The effectiveness of the SMF approach has been exemplified on datasets of increasing difficulty. In all cases, active learning unleashed the full potential of coupling the most confident and the most uncertain heuristics, which is the cornerstone of SMF. To our question about the need of adaptivity, and thus more complicated and fault-prone algorithms, the answer is unambiguous: a method versatile enough for accommodating various levels of difficulty on both the sensitivity and specificity criteria *must include on-line adaptivity* through active learning.

Future work will go further in the adaptivity direction. The first perspective considers self-calibrating not only the SMF balancing parameter, but also the SMFA parameters in an auto-learning approach. The samples are selected with two

strategies: the most uncertain predictions in the last run guide the selection of samples to enhance the current prediction confidence, while the random sampling strategy avoids overfitting the past.The current sample ratio between the two strategies is fixed and set to 1 : 1. However, a straightforward extension is to address this problem under the sequential decision optimization framework. The hybrid optimization indicator of [34] can be considered to efficiently balance the exploitation and exploration trade-off.

We have seen that performance was limited by the occurrence of abrupt changes, where the advantage of taking into account the past turns into a liability. The second perspective considers a semi-supervised online change detection framework that has already proved to be efficient at the level of the individual timeserie [32]. The next step would be to extend it towards the full matrix data.

ACKNOWLEDGMENTS

D. Feng acknowledges the support of the Chinese Scholarship Council. C. Germain acknowledges the support of the French cooperative project TIMCO, Pôle de Compétitivité Systematic (FUI 13). The authors acknowledge the support of The European Infrastructure Project EGI-InsPIRE INFSO- RI-261323 and France Grilles for providing computing resources on the French National Grid Infrastructure. The authors thank Michèle Sebag for stimulating discussions and Tristan Glatard for expert grid advice.

REFERENCES

[1] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *SIGACT News*, vol. 33, no. 2, pp. 51–59, 2002.

[2] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *14th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, 2008, pp. 426–434.

[3] E. Torres, G. Molto, D. Segrelles, and I. Blanquer, "A replicated information system to enable dynamic collaborations in the grid," *Concurr. Comput. : Pract. Exper.*, vol. 24, no. 14, pp. 1668–1683, 2012.

[4] S. Venugopal, R. Buyya, and K. Ramamohanarao, "A taxonomy of data grids for distributed data sharing, management, and processing," *ACM Comput. Surv.*, vol. 38, no. 1, Jun. 2006.

[5] W. Zhou, "Fault management in distributed systems," University of Pennsylvania Department of Computer and Information Science, Tech. Rep. MS-CIS-10-03, 2010.

[6] I. Rish, M. Brodie, S. Ma, N. Odintsova, A. Beygelzimer, G. Grabarnik, and K. Hernandez, "Adaptive diagnosis in distributed systems," *IEEE Trans. Neural Networks*, vol. 16, no. 5, pp. 1088 – 1109, 2005.

[7] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier, "Using magpie for request extraction and workload modelling." in *OSDI*, vol. 4, 2004, pp. 18–18.

[8] M. Y. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer, "Pinpoint: Problem determination in large, dynamic internet services," in *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*. IEEE, 2002, pp. 595–604.

[9] P. Reynolds, C. E. Killian, J. L. Wiener, J. C. Mogul, M. A. Shah, and A. Vahdat, "Pip: Detecting the unexpected in distributed systems." in *NSDI*, vol. 6, 2006, pp. 115–128.

[10] R. Fonseca, G. Porter, R. H. Katz, S. Shenker, and I. Stoica, "X-trace: A pervasive network tracing framework," in *Proceedings of the 4th USENIX conference on Networked systems design & implementation*. USENIX Association, 2007, pp. 20–20.

[11] X. Liu, Z. Guo, X. Wang, F. Chen, X. Lian, J. Tang, M. Wu, M. F. Kaashoek, and Z. Zhang, "D3s: Debugging deployed distributed systems." in *NSDI*, vol. 8, 2008, pp. 423–437.

[12] D. Geels, G. Altekar, P. Maniatis, T. Roscoe, and I. Stoica, "Friday: Global comprehension for distributed replay." in *NSDI*, vol. 7, 2007, pp. 285–298.

[13] X. Liu, W. Lin, A. Pan, and Z. Zhang, "Wids checker: Combating bugs in distributed systems." in *NSDI*, 2007.

[14] C. Killian, J. W. Anderson, R. Jhala, and A. Vahdat, "Life, death, and the critical transition: Finding liveness bugs in systems code," *NSDI 07: Networked Systems Design and Implementation*, pp. 243–256, 2007.

[15] A. Quiroz, M. Parashar, N. Gnanasambandam, and N. Sharma, "Design and evaluation of decentralized online clustering," *ACM Trans. Auton. Adapt. Syst.*, vol. 7, no. 3, pp. 34:1–34:31, 2012.

[16] X. Zhang, C. Furtlehner, C. Germain-Renaud, and M. Sebag, "Data stream clustering with affinity propagation," *IEEE Transactions on Knowledge and Data Engineering*, 2014.

[17] D. Feng, C. Germain-Renaud, and T. Glatard, "Efficient distributed monitoring with active collaborative prediction," *Future Generation Computer Systems*, vol. 29, no. 8, pp. 2272–2283, 2013.

[18] I. Rish and G. Tesauro, "Estimating end-to-end performance by collaborative prediction with active sampling," in *Integrated Network Management*, 2007, pp. 294–303.

[19] E. J. Candès and T. Tao, "The power of convex relaxation: Near-optimal matrix completion," *IEEE Trans. Inf. Theor.*, vol. 56, no. 5, pp. 2053–2080, 2010.

[20] B. Recht, "A simpler approach to matrix completion," *J. Mach. Learn. Res.*, vol. 12, pp. 3413–3430, 2011.

[21] N. Srebro, J. D. M. Rennie, and T. S. Jaakola, "Maximum-margin matrix factorization," in *Advances in Neural Information Processing Systems 17*, 2005, pp. 1329–1336.

[22] E. Laure and al, "Programming the Grid with gLite," pp. 33–45, 2006.

[23] I. Foster, "The globus toolkit for grid computing," in *IEEE Int. Symp. on Cluster Computing and the Grid*, 2001.

[24] M. Ellert and al., "Advanced resource connector middleware for lightweight computational grids," *Future Generation Computer Systems*, vol. 23, no. 2, pp. 219 – 240, 2007.

[25] A. Tsaregorodtsev and al., "DIRAC3 . The New Generation of the LHCb Grid Software," *Journal of Physics: Conference Series*, vol. 219, no. 6, p. 062029, 2009.

[26] J. T. Moscicki, "Diane - distributed analysis environment for grid-enabled simulation and analysis of physics data," in *Nuclear Science Symposium Conference Record, 2003 IEEE*, vol. 3, 2003, pp. 1617–1620 Vol.3.

[27] S. Bagnasco and al., "Alien: Alice environment on the grid," *Journal of Physics: Conference Series*, vol. 119, no. 6, p. 062012, 2008.

[28] T. Maeno, "Panda: distributed production and distributed analysis system for atlas," *Journal of Physics: Conference Series*, vol. 119, no. 6, p. 062036, 2008.

[29] S. Andreozzi and al., "Glue Schema Specification, V.2.0," http://www.ogf.org/documents/GFD.147.pdf, Open Grid Forum, Tech. Rep., 2009.

[30] T. Joachims, "A support vector method for multivariate performance measures," in *International Conference on Machine Learning (ICML)*, 2005, pp. 377–384.

[31] C. Scott, "Performance measures for neyman-pearson classification," *IEEE Trans. Inf. Theor.*, vol. 53, no. 8, pp. 2852–2863, Aug. 2007.

[32] D. Feng, "Efficient end-to-end monitoring for fault management in distributed systems," Ph.D. dissertation, Universite Paris Sud, 2014.

[33] J. D. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 713–719.

[34] W. Wang and M. Sebag, "Hypervolume indicator and dominance reward based multi-objective Monte-Carlo Tree Search," *Machine Learning*, vol. 92, no. 2-3, pp. 403–429, May 2013.