HAL
archives-ouvertes.fr

Non-Markovian Reinforcement Learning for Reactive Grid scheduling

Julien Perez, Balázs Kégl, Cecile Germain-Renaud

Non-Markovian Reinforcement Learning for Reactive Grid scheduling

Julien Perez[1], Balazs Kegl[2], Cecile Germain Renaud[3]

[1] Telecom ParisTech,
Département TSI, UMR CNRS 5141,
F-75634 Paris Cedex 13, France
`julien.perez@telecom-paristech.fr`

[2] Université de Paris-Sud,
Laboratoire de Recherche en Informatique, Bâtiment 490,
F-91405 Orsay Cedex, France
`cecile.germain@lri.fr`

[3] Université de Paris-Sud,
Laboratoire de l'Accélérateur Linéaire,
F-91405 Orsay Cedex, France
`balazs.kegl@lal.in2p3.fr`

Résumé :
Two recurrent questions often appear when solving numerous real world policy search problems. First, the variables defining the so called Markov Decision Process are often continuous, that leads to the necessity for discretization of the considered state/action space or the use of a regression model, often non-linear, to approach the Q-function needed in the reinforcement learning paradigm. Second, the markovian hypothesis is made which is often strongly discutable and can lead to unacceptably suboptimal resulting policies. In this paper, the job scheduling problem in grid infrastructure is modeled as a continuous action-state space, multi-objective reinforcement learning problem, under realistic assumptions ; the high level goals of users, administrators, and shareholders are captured through simple utility functions. So, formalizing the problem as a partially observable Markov decision process (POMDP), we detail the algorithm of fitted Q-function learning using an Echo State Network. The experiment, conducted on simulation of real grid activity will demonstrate the significative gain of the method against native scheduling infrastructure and a classic feed forward back-propagated neural network (FFNN) for Q function learning in the most difficult cases.

1. Introduction

This paper proposes an approach that uses Reinforcement Learning (RL) as a scheduling (resource allocation) mechanism. In the following, this function (scheduling) will be called *supervision*, and the corresponding software entity the *supervisor*. The flexibility of an RL-based system allows us to model the state of the resources, the jobs to be scheduled, and the high-level objectives of the various grid actors. RL-based supervision can seamlessly adapt its decisions to changes in the distributions of inter-arrival time, QoS requirements, and resource availability. Moreover, it requires minimal prior knowledge about the target environment including user requests and infrastructure.

We develop a general RL framework with models for classes of jobs (currently two, best-effort and responsive), for objective functions, and for the infrastructure. Furthermore, by exploiting recent advances, Jaeger (2003), in approximating the value function through recurrent neural networks, we are able to relax the Markovian condition often unrealistic due to lacks of description of the decision process. This work has been developed in the framework of the flagship EU grid infrastructure EGEE (Enabling Grid for E-SciencE) both for the grid model, and for the experimental data. The major contributions of our paper are as follows. First, we describe a formalization of the supervision problem as a partially observable continuous action-state space, multi-objective reinforcement learning problem, under realistic hypotheses. Second, we explore implementations of the reinforcement learning framework integrating those high level goals. Third, we show experimentally that our RL-based supervisor achieves responsiveness without degrading utilization, as measured by several metrics related to user and administrator satisfaction. We finally show that in difficult cases, memory enhanced models, like Echo State Network, offer better results.

2. Grid scheduling

2.1. Responsiveness requirement

Major industry players acknowledge interactivity as a critical requirement for enlarging the scope of high performance computing, Mirman (2006), and invest in this direction. Nonetheless, the general vision remains that large to massive computations dominate the e-science workloads. It might be consi-
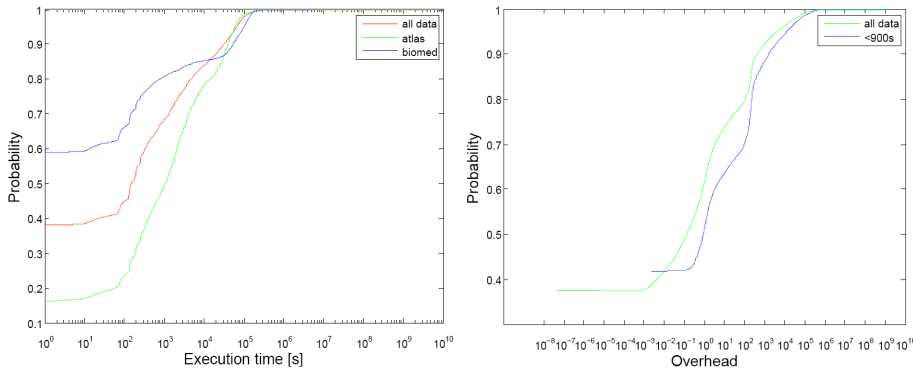
FIGURE 1: Cumulative distribution of execution times in the EGEE grid.(left)
Distribution of the relative overhead. (right)

dered as a self-realizing prediction : a long-latency software infrastructure has little appeal for tasks requiring responsiveness. The reality is more complex : because the resources and the data are there and because the workflows include long and short computations, users requiring responsiveness have little choice and do exploit the unresponsive infrastructure, albeit with repeated requirements towards improving QoS. Considering it has tens of thousands of CPU's, petabytes of storage, an extensive coverage of scientific communities, and the perspective of sustainable development, the EGEE grid provides a good approximation of the current needs of e-science. With extensive monitoring facilities already in place, EGEE offers an unprecedented opportunity to observe and gain understanding of new computing practices of e-science. We analyze here more than one year of EGEE production under gLite, Laure & al (2006), the EGEE middleware. The trace was provided by the Real Time Monitor project, Colling & McGough (2006). The trace covers the period November 2005 to January 2007 and includes more than 17 million production jobs belonging to 114 virtual organizations (VOs). Jobs launched by operations management for testing service availability have been removed from the trace, thus the results faithfully describe user activity.

The left part of the Fig. 1 shows the distribution of execution times from this trace. The striking feature is the importance of short jobs. All requests aggregated, the 70% quantile is approximately 900 seconds. These data also support our claim that all scientific communities need responsiveness. This is obvious for the biomedical community (Biomed VO), with more than 80% of short jobs. However, even the Atlas VO (the largest HEP community) features

more than 50% of short jobs. The right part of the Fig. 1 shows the distribution of V, the dimensionless *relative overhead* ; V is the ratio of the time spent in queue to the execution time of a job ; the time spent in the middleware stack is not included, thus the relative overhead is only related to the supervision issue addressed in this paper. 40% of the short jobs experience a tenfold slowdown due to queuing delays alone, a clear indicator of un-responsiveness.

2.2. Configuration-based Responsiveness

The responsiveness is a major requirement of many of the mainstream users of the EGEE grid (HEP, Biomedical), Blanchet *et al.* (2006). Nonetheless, only a handful of sites have actually deployed existing solutions. The explanation lies in the "expected return of investment" for site administrators. Enabling a Short Deadline Job (SDJ) service requires modifying the configuration files of the local batch schedulers, which have been carefully crafted and stabilized to answer users and institutions requirements related to access rights and shares. The lack of responsiveness together with the independent problem of fault management presently leads to an increasing usage of overlay schedulers, Mościcki *et al.* (2007), exploiting placeholder jobs ("pilot jobs") in EGEE, which is at the same time recognized as an unsustainable solution with respect to both manpower and resource provisioning. An interesting by-product of this development is the consensus of the community on what level of Quality of Service could reasonably be expected. Some middleware penalty is the unavoidable counterpart of a large scale system ; a typical delay of 2 minutes is thus considered acceptable, meaning that only jobs that are really fine grained should be compelled to resort to overlay schedulers.

3. Non-Markovian RL Framework

This section describes the RL models of the supervision problem. For the sake of completeness, the first section briefly recalls the basics of Markov Decision Process (MDP) and RL.

3.1. Markov Decision Process

A Markov Decision Process is a quadruple (S, A, P, R) where S is the set of possible *states* of the system, A is the set of *actions* (or decisions) that can be taken, and P is a collection of *transition probabilities* $P_{ss'}^a = P\{s_{t+1} =$

$s'|s_t = s, a_t = a\}$ that map the current state and action to the next state. The function $R^a_{s,s'} : S \times A \times S \rightarrow \mathbb{R}$ defines the *rewards* earned when moving from state s to state s' through action a. The goal is to find a stationary policy $\pi^* : S \rightarrow A$ which chooses the action to take in each state, without knowledge of the past history (other than what is summarized in the state). The objective is to maximize the the long-term expectation of the rewards, the so-called *value function*

$$Q^\pi(s, a) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right]$$

where $\gamma \in [0, 1]$ is a discount factor dampening the future rewards. In the scheduling context, P and R (the environment dynamics) are unknown, so the Q function has to approximated through repeated experiments. This is the definition of reinforcement learning, Sutton & Barto (1998) : the optimal policy will be learned by interactions with the environment. The markovian propertie is an essential characteristic of an MDP because the reward and the transition primitives are supposed to depend only on the current state and the choosen action, and not on the historic of the system (e.g. the past choosen actions and states). Formally, an historic h_t is a finite sequence of state/action pairs : $h_t = (s_0, a_0, ..., s_{t-1}, a_{t-1})$. Let H_t, the set of all historics at the time t, the markovian property is defined by : $\forall t, \forall h_t \in H_t, \forall s_t, s_{t+1} \in S, \forall a_t \in A$,

$$P(s_{t+1}|h_t, s_t, a_t) = P(s_{t+1}|s_t, a_t) = p(s_{t+1}|s_t, a_t)$$

Concerning this last point, a memory mecanism will be necessary if the markovian property of the considered decision process is not admit. In this case, the reward function will not only depend on the current state but also on the serie of the past states, encountered by the system. If we consider a decision problem in a complex environment, defining markovian decision process correspond to admit that we are able to define a suffisantly complete description of the environment to resume the past of the simulation. When this definition is not verified, the past states of the environment must be included as inputs of the decision function of the agent. A famous illustration of this phenomen is presented in, Tolman (1932). In the next section, we propose a novel algorithm of Q-function presented as a memory enhanced adaptation of the Fitted Q-Iteration algorithm, Ernst *et al.* (2005).

3.2. Echo State Network based Fitted Q Iteration (ESN-FQ)

The basic idea underlying any fitted Q iteration based algorithm, Ernst *et al.* (2005); Antos *et al.* (2007), is the following : instead of updating the value function on-line, the update is performed off-line considering an entire set of transition experiences. Experiences are collected in triplets of the form (s, a, s') by interacting with the (real or simulated) system. Here, s is the original state, a is the chosen action and s' is the resulting state. The set of experiences is called the sample set D. The consideration of the entire training information instead of on-line samples, has an important consequence : It allows the application of advanced supervised learning methods, that converge faster and more reliably than online gradient descent methods. In the case of an ESN-based regression of the Q-function, as proposed in this paper, the arrival order of the triplets must be preserved to use the memory capability of the model.

Algorithm 1 Echo State network based Fitted Q Iteration

Require: a set of transition samples D

1: $k = 0$
2: $Q_0 \leftarrow \text{init\_ESN}()$
3: **while** $k < N$ **do**
4:      generate_pattern_set, $\text{P} \leftarrow \{(\text{input}^l, \text{target}^l), l = 1, ..., D\}$ where :
         $\text{input}^l \leftarrow s^l, a^l,$
         $\text{target}^l \leftarrow c(s^l, a^l, s'^l) + \gamma \min_b Q_k(s'^l, b)$
5:      $Q_{k+1} \leftarrow \text{ESN\_training}(\text{P})$
6:      $k \leftarrow k + 1$
7: **end while**
8: **return** Q-value function Q_N

The algorithm is presented in algo. 1. The *init_ESN* and *ESN_training* procedures refer to ESN learning methodology proposed in Jaeger (2003). The *generate_pattern_set* procedure refers to the computation of the corresponding reward for each (state,action) couple from the simulation in the environment. The algorithm is an adaptation of, Riedmiller (2005). This procedure aims to efficiently regress a Q-function model and taking into account the memory of the considered decision process by using an ESN as a learner.

3.3. The Supervision Model

As explained before, any reinforcement learning formalization needs to define states, actions, and rewards for a given problem. We propose a set of variables describing states and actions to allow the formulation of the grid scheduling problem and the resource provisioning problems as continuous action-state space reinforcement learning problems.

State Space : the Grid Model

A complete model of the grid would include a detailed description of each queue and of all the resources. This would be both inadequate to the MDP framework and unrealistic : the dimension of the state space would become very large. Instead, the state is represented by a the following set of real-valued variables : (1) the total workload of running jobs ; (2) the time before a resource becomes available ; (3) the backlog, that is, the amount of work corresponding to queued jobs ; (4) the number of idle machines ; (5) the proportion of jobs of each VO in the queues. Any job management system provides the last two descriptors at any time. The three first descriptors, associated with workload, are discussed below.

Action Space : the Job Model

Each waiting job is a potential action to be chosen by the scheduler. As a consequence, except if there is no job waiting, the scheduler will always select a job when a resource become available (*greedy* allocation). A job is represented by a set of descriptors : the type of the job (batch/interactive) ; the VO of the user who submitted the job and the execution time of the job (the time to complete the job without any queuing or management overhead).

The VO associated with the job is a mandatory feature in large scale grids systems, and is available along the whole lifecycle of the job. If the choice between batch and interactive quality of service is proposed by the grid environment, the knowledge of this type is a realistic assumption : the interactive/batch flag is known for each job before the execution, and since the user has a strong interest to correctly specify it, we can trust it.

Workload Descriptors

The first three descriptors in the state space and the last one in the action space are related to the execution time of jobs. In this work, we compare two models. The *oracle* (ORA) model assumes perfect knowledge of this quantity when the job is placed in the queue. Although utterly unrealistic (except in very specific cases), this hypothesis provides an upper bound on the quality of the RL-based scheduling and resource provisioning as well.

The *estimated* (EST) model makes a very crude estimate of the execution time by the median of the execution times for the batch and interactive jobs, respectively, along an extended time window. Here, the hypothesis is fully realistic : historical data are readily available online inside gLite.

Rewards

The reward function proposed is a combination of the *responsiveness utility* and the *fairness*.

The responsiveness utility, W_j, is formally defined, for a job j, as

$$W_j = \frac{\text{execution time}_j}{\text{execution time}_j + \text{waiting time}_j}$$

It represents the reward associated with minimizing the overhead, V_j, ($W_j = 1/(1 + V_j)$). In both the *oracle* and *estimated* models, the rewards are computed when the job completes, thus when its actual execution time and queuing delays are available. Hence, the delay separating the action and the reward is highly variable ; with their short execution time, interactive jobs have a more immediate impact on the learning process.

The fairness represents the difference between the actual resource allocation and the externally defined share given to each VO. The allocation process should be such that the service received by each VO is proportional to this share. If there are n VO's, the shares are usually expressed as the n-vector of the percentages of the total resources $w = (w_1, \ldots w_n)$. Let S_{kj} be the fraction of the total service received by VO k up to the election of job j. Then, the deficit distance between the optimal allocation and the actual allocation is a good measure of the unfairness. The deficit distance is defined as

$$D_j = \max_k (w_k - S_{kj})_+$$

where $x_+ = x$ if $x > 0$ and 0 otherwise. The unfairness is bounded above by $M = \max_k(w_k)$. A normalized fairness reward can thus be derived by a simple linear transform. If M is the maximal unfairness, the fairness utility F_j associated to the election of job j is

$$F_j = -\frac{D_j}{M} + 1$$

With these definitions, the reward is in the $[0, 1]$ range, thus on the same scale, and is defined as

$$R_s = \lambda_s W + (1 - \lambda_s)F$$

4. Experimental Setup

We developed a simulation framework to evaluate the performance of RL-based scheduling. This section presents the simulation methodology, the workloads, and the experiments.

4.1. Simulation Methodology

We perform a discrete event simulation of the complete lifecycle of jobs. The events are submission, dispatch, and termination. The submission of a job adds an entry onto a shared queue. The state of the art batch schedulers heavily rely on multiple queues, with a simple FIFO scheduling inside each queue and prioritization amongst queues based on configuration files. Although our simulator can manage multiple queues, one of the goals of this work is to show that model-free methods are more effective, thus all jobs are fed to a unique queue. The core of the simulator is the learner. The reservoir of the ESN is composed by a set of 100 sigmoidal neurons, the weights are randomly fixed in $[0, 1]$ with 10% of connectivity between the neurons of the reservoir and 15% of connectivity between the reservoir and the output neurons as in, Jaeger (2003). In all simulations, the first and last 500 jobs were dropped from the result, in order to avoid the bias in the results introduced by the ramp-up and draining phases.

4.2. The Input Workload

The basis for the input workload is a trace from EGEE, namely the log of the PBS scheduler of the LAL site of EGEE. The trace covers the acti-

	Size	Mean	Median
Interactive	4480	108 ± 190	160
Batch	5020	34927 ± 78755	15615

TABLE 1: The EGEE workload. *Size* is the number of jobs. The statistics refer to the execution time. All times are in seconds.

vity of more than seven weeks (from 25 Jul 2006 to 19 Oct 2006). It includes more than 9000 user jobs, not counting the monitoring jobs which are executed concurrently with the user jobs and consume virtually no resource ; they were removed from the trace. All jobs are sequential, meaning that they request only one core. From this trace we had to decide which requests are tagged as either interactive or batch, in order to simulate a situation where such requirement for QoS would be proposed. While the submission queue could have provided some hint, most queues include jobs with the full range of execution times. This is due to the fact that the queues are mostly organized along VO's, not along quality of service. We decided to tag jobs with an execution time less than 900 seconds as interactive jobs, and the other ones as batch jobs. Otherwise, the workload is kept unchanged. Table 1 summarizes the characteristics of the trace. The identifier of the target resource which is described in the PBS log as a core is included in the trace. In the period use in the experiments, the number of available cores is fairly constant ($P = 81$). The extended timescale of the trace offers the opportunity to test the capacity of the RL-based supervisor to adapt to changing conditions. The large value of the standard deviation in Table 1 is a first indicator of high variability. Amongst the VOs present in the trace, only four contributed significantly. The target vector is $(0.53, 0.19, 0.1, 0.18)$. The last share corresponds to the aggregation of the small VOs. In the segment considered in the workload, the fairness utility of the native scheduler is nearly constant (after the ramp-up phase) at 0.7.

4.3. The Experiments

We ran simulations using the workload described above with the following configurations. The resource configuration is rigid ; the number of cores P is fixed (to 81, for comparison with EGEE). Thus, we experiment :
- ORA - The actual execution times are assumed to be known at the submission time (*oracle* model).

- EST - The execution times are estimated by the median of their respective categories (*estimated* model).

Inside this setting, the weight λ_s of the responsiveness utility is set to 0.5. We compare our results, using ESN and standard FFNN with the NAT experiment, which is the result of the activity of the EGEE scheduler as collected in the trace.

5. Experimental Results

The most important performance indicators are related to 1) the performance of the RL method itself, and 2) the satisfaction of the grid actors. The quality of the optimization performed by the RL is measured by the distribution of the target indicator, which is the responsiveness utility W. Even if W can be satisfactorily optimized, it remains prove that it correctly captures the users' expectations regarding QoS. The user experience is dominated by the wallclock queuing time, which is also reported. Considering fair-share, we report the difference between the fair-share achieved by the native scheduler (as the state of the art for fair-share), and the fair-share of our scheduler ; both are computed following equations defined in section 3.3..

5.1. Responsiveness

Table 2 presents the summary statistics for the responsiveness W, and Fig. 2 shows the inverse cumulative distribution functions of W (*i.e.* $P(W > x)$ as a function of x). The first result is that the RL architecture (including the ESN) efficiently optimizes the responsiveness objective. Recall that from the definition of W, the closer W is to 1, the better. Considering the summary statistics, the average responsiveness of the RL-based methods applied to interactive jobs is typically 0.90, while the native scheduler achieves only 0.63. Moreover, the standard deviation is reduced by approximately 50%.

For batch jobs, the RL-scheduler does not degrade the average performances, and there is even a slight improvment. Considering the distribution (Fig. 2), W is larger than 0.9 (that is, off the optimum by 10% or less) more than 90% or more of the interactive jobs. The plots have been truncated on the vertical axis for readability. The right figure of Fig. 2, that only depicts results for RL-based methods, shows that ESN-based RL approaches allow to improve results for the 5% worth cases of interactive jobs that encounter a responsiveness perform of less that 0.4.

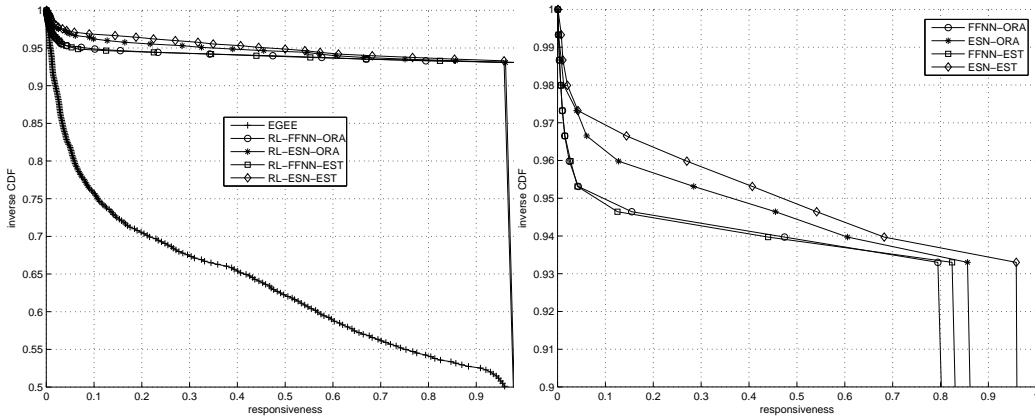| Algorithm	Batch	Interative	Total
EGEE	0.82 ± 0.26	0.62 ± 0.41	0.73 ± 0.36
RL-NN-ORA	0.93 ± 0.18	0.94 ± 0.22	0.93 ± 0.20
RL-NN-EST	0.92 ± 0.18	0.94 ± 0.22	0.93 ± 0.20
RL-ESN-ORA	0.93 ± 0.19	0.94 ± 0.20	0.93 ± 0.19
RL-ESN-EST	0.93 ± 0.17	0.95 ± 0.19	0.94 ± 0.18

TABLE 2: Responsiveness statistics

FIGURE 2: Inverse CDF of responsiveness for interactive jobs

The second result is that switching from the unrealistic oracle setting to a simple estimation method does not degrade the performances. A explanation could be that using an estimation of the execution time eliminates noise only present in real data and allow to improve error minimization during Q function learning. Turning to the comparison with the native scheduler, Table 2 and Fig. 2 show that the RL scheduler improves massively the native scheduler for all the jobs (both interactive and batch). For the interactive case, only 53% of the jobs reach a $0.5\ W$ in the native scheduler, versus more than 90% in the RL scheduler. The superior performance of interactive jobs tends to prove that the responsiveness was indeed a good optimization target.

5.2. Queuing Delay

We now consider the queuing delay (Table 3 and Fig. 3). A first observation concerns the variance of the results that seems to be essentially due to the typical bursty distribution of the workload. In this context, ESN-based RL methods seem to offer better behaviours for this indicator, which is consistent with the observations of the last section.

Algorithme	Batch	Interactive
EGEE	13041 $\pm$26726	2756 $\pm$8844
RL-NN-ORA	3777 $\pm$12168	1366 $\pm$7989
RL-NN-EST	3803 $\pm$12232	1382 $\pm$8076
RL-ESN-ORA	3962 $\pm$13014	1085 $\pm$8491
RL-ESN-EST	3492 $\pm$12175	495 $\pm$3589

TABLE 3: Queuing delays statistics, in seconds

As discussed in section 2.2., the 2 minutes delay is a good landmark for assessing the potential satisfaction (or lack thereof) of the interactive users. Under the RL-based scheduler, more than 90% of the interactive jobs experiment a queuing delay below 2 minutes compared to the 63% featured by EGEE. Since the RL-based models keep slightly less than 4% of the jobs above 2 minutes, the limits might be in the model itself. More precisely, the alternatives are either that the exploration/exploitation parameter is to be increased, or that the relatively low frequency (each 15 minutes) of machine pool evolution limits the capacity to adapt the resources to bursts. We are currently trying to characterize more precisely the configurations of the outliers, including the arrival and request processes, and the internal dynamics of both the scheduling and provisioning MDP in response to them.

5.3. Fairness

Fig. 4 shows the dynamics of the fair-share. The horizontal axis is the simulated time, and the vertical axis is the difference between the RL and EGEE fairness utilities. The difference is actually negligible. Most of the time, the RL schedulers are marginally superior, with some excursions corresponding to bursts.
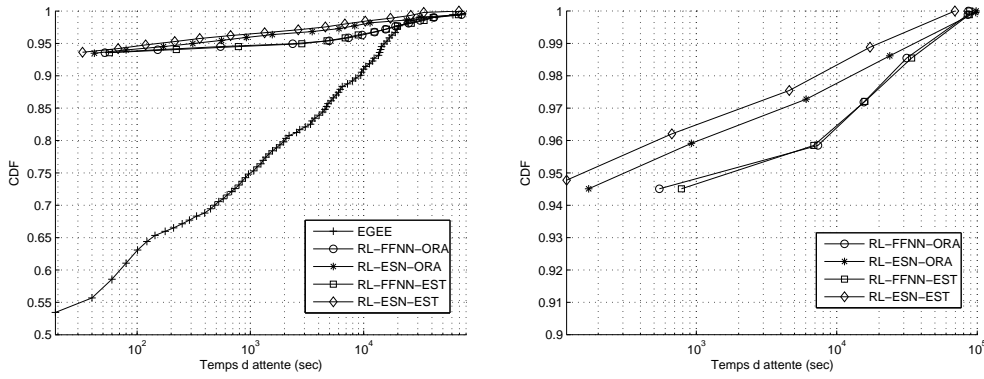
FIGURE 3: Distribution of the queuing delay for interactive jobs. All times in seconds.
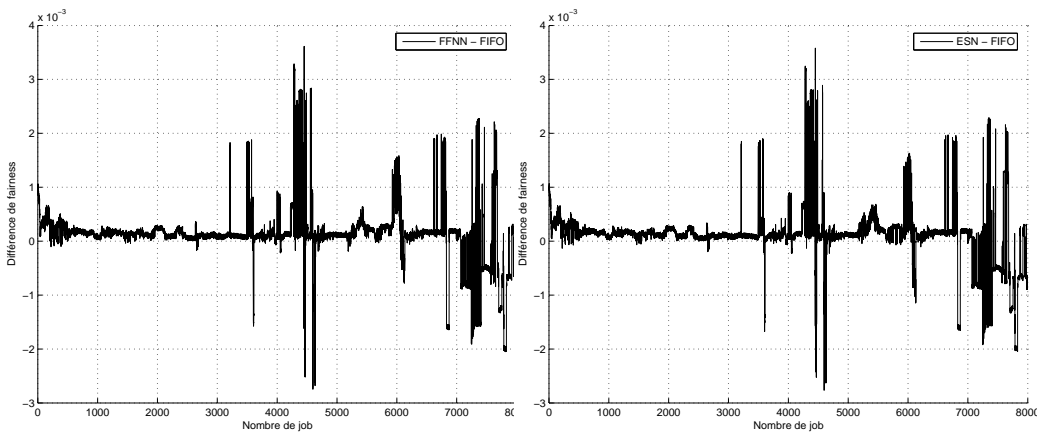

FIGURE 4: Dynamics of the fair-share

6. Conclusion and Perspectives

This paper deals with a problem that exemplifies a real-world situation where traditional, configuration-based solutions reach their limits, calling for autonomic methods. In term of machine learning, a first interesting result is the capability of an ESN based Fitted-Q iteration algorithm to produce an efficient scheduling policy, taking into account the memory of the considered decision process. A second interesting result is the robustness of the method to crude estimations, in a situation where the variability is high.

In term of autonomic computing, our future work will follow two avenues. The first one will integrate a more refined model of the switching delays, based on realistic hypothesis of future grid-over-clouds deployments. The second one will explore more aggressive methods for favoring interactive jobs when the RL-based supervision appears to be lagging behind.

Références

ANTOS A., MUNOS R. & SZEPESVÁRI C. (2007). Fitted Q-iteration in continuous action-space MDPs. In *NIPS* : MIT Press.

BLANCHET C., MOLLON R., THAIN D. & DELEAGE G. (2006). Grid Deployment of Legacy Bioinformatics Applications with Transparent Data Access. In *7th IEEE/ACM International Conference on Grid computing*, p. 120–127.

COLLING D. & MCGOUGH A. (2006). The gridcc project. In *International Conference on Communication System Software and Middleware*, p. 1–4.

ERNST D., GEURTS P. & WEHENKEL L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, **6**, 503–556.

JAEGER H. (2003). Adaptive nonlinear system identification with Echo State Networks. In *Advances in Neural Information Processing Systems 15*, p. 593–600 : MIT Press.

LAURE E. & AL (2006). Programming the Grid with gLite.

MIRMAN I. (2006). Going parallel the new way.

MOŚCICKI J., BUBAK M., LEE H., MURARU A. & SLOOT P. (2007). Quality of service on the grid with user level scheduling. In *Cracow Grid Workshop*, p. 119–129.

RIEDMILLER M. (2005). Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method. In *Machine Learning : ECML 2005, 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005, Proceedings*, volume 3720 of *Lecture Notes in Computer Science*, p. 317–328 : Springer.

SUTTON R. S. & BARTO A. G. (1998). *Reinforcement Learning : An Introduction*. MIT Press.

TOLMAN E. C. (1932). Purposive behavior in animals and men. *The Psychoanalytic review*.