# A Performance Prediction Module for Workflow Scheduling

David Monge[1,4], Jiří Bělohradský[3], Carlos García Garino[1,2] Filip Železný[3]

[1] ITIC, Universidad Nacional de Cuyo, Mendoza, Argentina
[2] Facultad de Ingeniería, Universidad Nacional de Cuyo, Mendoza, Argentina
[3] Czech Technical University, Prague, Czech Republic
[4] PhD Fellowship CONICET
{dmonge, cgarcia}@itu.uncu.edu.ar {belohjil,zelezny}@fel.cvut.cz

**Abstract.** Through the years, scientific applications have demanded more powerful and sophisticated computing environments and management techniques. Workflows facilitated the design and management of scientific applications. The complexity of today's workflows demand a high amount of resources and mechanisms for provisioning them. The execution of scientific workflow applications is a complex task and depends on how the resources are assigned. Scheduling is the name given to the process that assigns computing resources to the tasks comprised in a workflow. This work presents a scheduling algorithm (PPSA) for workflows tightly coupled to a performance prediction module (PEM). A set of experiments was developed for measuring the performance of the algorithm using the information provided by the proposed performance module. The proposed algorithm is compared with an algorithm included in the well-known workflow middlewares Condor DAGMan and ASKALON.

**Keywords:** Workflow, Scheduling, Performance Prediction, Nonparametric Regression

## 1 Introduction

In the last 30 years approximately, engineering and scientific areas have been using computation as a main component in their experimentation and simulation processes. At the beginning, supercomputers were used to simulate large complex systems composed by partial differential equations. These early developments brought with them the need of generating tools for visualization and data analysis. The complexity of such applications grew and thus more sophisticated methods for data movement and storage were required. Such tasks began to be repeated often and then scripts for the automation of the job were written. Such scripts started to be more complicated and the tasks to execute demanded more resources. Soon, the distribution of the workload across interconnected computers was needed. The development of distributed computing technologies allowed to deal with the execution of remote jobs and data. Workflow applications were introduced to scientific areas due to the flexibility they provide for design and management of applications.

Today's applications require a large number of resources and management software. The complexity of actual workflow applications require a vast amount of resources. Such requirements can be fulfilled by using computing infrastructures deployed across organizations or in larger scale platforms such as the Grid. The high heterogeneity of such kind of infrastructures introduces challenges that must be addressed for a proper workflow applications management. The process of resource assignment is central for the the efficient execution of workflow applications. Mentioned process is known in the literature as workflow scheduling. A big number of solutions to workflow scheduling has been developed and there are still some improvement possibilities to actual approaches.

This paper is structured as follows. The concepts related with workflow scheduling are introduced in next section. Section 3 describes scheduling scheme. Section 1 describes the scheduling algorithm incorporated in the DAGMan and ASKALON workflow systems. Section 5 discusses the experiments designed and the obtained results. Finally, section 6 presents conclusions and future work.

## 2 Background

A workflow can be defined as a set of tasks that must be performed in certain order for achieving a particular objective. Formally a workflow can be defined as $\varUpsilon(\varGamma, \varDelta)$, where $\varGamma$ is the set of tasks comprised in the workflow and $\varDelta$ the set of dependencies between pairs of tasks. There is a typical classification for workflows, according to their structure: *Directed Acyclic Graph (DAG)* and *Non-DAG* workflows [1].

DAG workflows do not allow cycles in their structure. In opposition, Non-DAG workflows allow the repetition of tasks or sequences of them by including cycles in their structure. However, DAG workflows are suitable for representing most of the required applications in science and engineering research areas. Thus, the major part of the efforts have been focused on DAG workflows [2,3].

Workflow Scheduling is the process in which the selection of resources for the tasks is performed according to a specific scheduling objective to met [2,3]. Scheduling is a complex process that is tightly coupled with other tasks such as resource discovery, matchmaking, fault tolerance, information retrieval and so on. Information retrieval is probably one of the most important features required for workflow scheduling. The availability of accurate and updated information is crucial for the proper performance of schedulers.

Schedulers focused on minimization of the execution time (makespan minimization) require information about the performance of the workflow tasks and the transfer times between them. Then, it is crucial to know the *execution time* and *outputs size* of the tasks for the selection of the proper resources.

**Tasks execution time prediction** In general, tasks execution prediction can be solved by Code Analysis [4], Analytic Benchmarking/Code Profiling [5,6]

and Regression Techniques [7]. First two families of prediction methods require a deep knowledge of the tasks performance that is not always fulfilled by disciplinary users. In opposition, regression methods do not require such knowledge because they use previous execution data for the prediction of execution times.
There are two types of regression methods: *Parametric Regression* and *Nonparametric Regression*. Parametric regression techniques construct a prediction model by learning parameters of a function that describes the data. Main disadvantage of such kind of techniques is that in the presence of new information, the prediction model needs to be reconstructed. On the other side, nonparametric regression techniques do not require the construction of a prediction model. Such techniques are also known as data-driven because predictions are done based on the available data. When new performance information is acquired it can be added to the performance database for being used in future predictions. This feature allows the construction of an adaptive prediction scheme.

**Tasks output size prediction** To the best knowledge of the authors, no work in the prediction of tasks output size has been done. The availability of such information is crucial for the estimation of the transfer times between tasks. To address such issue, we consider that it is convenient the use of non parametric regression techniques based on the arguments discussed in the previous paragraphs.

There are a number of solutions for addressing performance prediction issues related with scheduling. Some of them are: Network Weather System (NWS) [8], Performance Analysis and Characterization Environment (PACE) [9], Grid Harvest Service (GHS) [10].

In a previous work [11], a workflow scheduling algorithm was presented. The performance information of such algorithm was provided by a simple performance model based on Analytic Benchmarking and Code Profiling suitable for a specific type of synthetical tasks. In this work, we propose a scheduling algorithm with performance estimation module for tasks execution time and outputs size prediction based on a nonparametric regression technique.

## 3 Workflow Scheduling based on Performance Prediction

This section discusses a novel scheduling algorithm for workflows in the context of Grid. Such algorithm uses a prediction module for retrieving necessary performance information. The scheduling algorithm is described in subsection 3.1. Subsection 3.2 presents the mentioned performance estimation module.

### 3.1 Performance Prediction Scheduling Algorithm

The Performance Prediction Scheduling Algorithm (PPSA) focuses on the obtainment of resource mappings that minimize the makespan of DAG-based workflows. PPSA is a Branch and Bound (BB)-based algorithm which presents non-polynomial complexity [12,13,11]. Although in general, algorithms with polynomial complexity are preferred over non-polynomial ones, PPSA has proven to be an adequate solution obtaining important workflow execution improvements with a minimal scheduling overhead [11]. Such algorithm is proper for medium size workflows with about 200 tasks.

For larger size workflows a partitioning method can be applied for the obtainment of manageable sub-workflows. Then, such sub-workflows can be scheduled by PPSA. A similar approach called *Workflow Partitioning and Deferred Planning,* is adopted by the Pegasus workflow system [14]. However, this work does not addresses the partitioning of workflows.

PPSA aims to reduce the workflows makespan based on performance estimations. The scheduling process is carried out by finding a solution to a Constraint Satisfaction and Optimization Problem (CSOP). PPSA receives as input a problem definition determined by the tuple $\langle X, D \rangle$ where $X$ is the set of workflow tasks (variables set) and $D$ corresponds to the set of compatible machines for each workflow tasks (domains set).

For the reduction of the characteristic algorithm's complexity, the problem definition is simplified in a Pre-assignment stage [11]. Such pre-assignment stage reduces the domains of the critical tasks in the workflow to the most powerful resources. If a task can not be delayed without an increase of the workflow makespan, then it is a critical task. The details of the scheduling algorithm can be seen in our previous works [11,12,13]. A schema of PPSA architecture and its interaction with the Prediction Module is presented on figure 1.

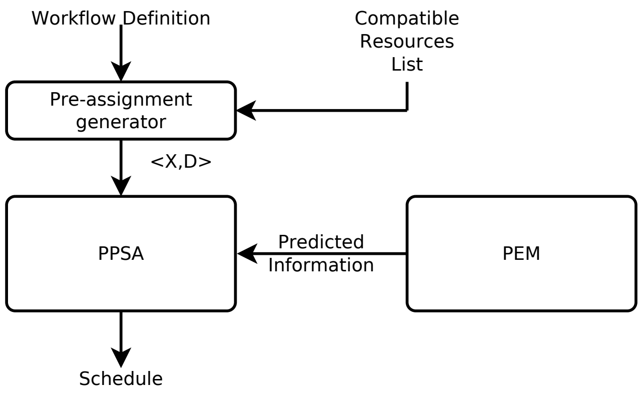


Fig. 1: PPSA architecture and interaction with the Performance Module.

For the applicability of PPSA on real environments, scheduling decisions must be taken based on realistic performance information. For that purpose we developed a Performance Estimation Module (PEM) that is in charge to provide accurate performance information to PPSA.

The information provided by the prediction module is required in two different times and contexts. All the information related with tasks outputs is predicted before PPSA is invoked. The prediction of output sizes depends only on the parameters and inputs provided to the tasks. And thus, the prediction of such information can be calculated independently of the PPSA execution. The predicted outputs size information is used by PPSA on the calculation of the transfer times between two tasks $A$ and $B$. The calculation of the transfer time between two tasks is calculated as:

$$T_{tx}(A, B) = PEM.outputSize(A)/TR_{machineA,machineB} \tag{1}$$

where $PEM.outputSize(A)$ is the output size of the task $A$ estimated by the prediction module, to be transmitted to task $B$. And $TR_{machineA,machineB}$ is the transfer rate between the resources assigned to the tasks $A$ and $B$.

In opposition to the output size information, the prediction of tasks execution time is performed during the execution of PPSA. This information is used by PPSA to decide if it is proper to assign the candidate resource to the evaluated task. The calculation of the execution time of a task on a resource is calculated as:

$$T_{ex}(A, R) = PEM.executionTime(A, R) \tag{2}$$

where $PEM.executionTime(A, R)$ is the predicted execution time of the task $A$ on the resource $R$.

Next subsection details the characteristics of the estimation module and how the performance prediction process is carried out.

## 3.2 Performance Estimation Module

For the achievement of good workflow execution performance, the scheduling decisions must be performed based on accurate tasks performance information. Such information is provided by the Performance Estimation Module (PEM) as was explained in the previous subsection. On figure 2 a schema of PEM is presented.

PEM uses the *k-nearest neighbors* (k-NN) technique [15], which belongs to the family of non parametric regression methods. The k-NN algorithm makes predictions based on the similarity of the instances included in the dataset. It is worth noting that nonparametric regression techniques may require a considerable execution time in the case that a large number of instances are considered in the prediction process. To address such issue, a method for keeping a manageable size of the performance dataset is required. However, such issue is not addressed in this work.

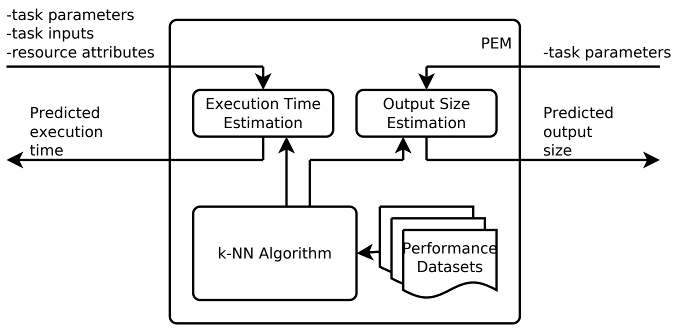


Fig. 2: Performance Module Architecture.

**Task Performance Dataset** For the maximization of the efficiency of the prediction method it is convenient to maintain separate datasets for each type of task. This is reasonable since the fact that components of workflows are reusable. In Figure 3 the header of the performance dataset for a task called `weka-kmeans`. The performance database is expressed in the Attribute Relation File Format (ARFF) [15].

```
@relation weka-kmeans-performance
@attribute in-dataset string
@attribute arg-numberOfClusters numeric
@attribute out-output-size numeric
@attribute mach-TotalCpus numeric
@attribute mach-CpuFrequency numeric
@attribute mach-JavaMFlops numeric
@attribute mach-TotalMemory numeric
@attribute time numeric
@data
```

Fig. 3: Header of the performance database of the Weka K-means Clustering task.

The ARFF file for the K-means task type maintains information about: *i)* the input of the task (`in-dataset`); *ii)* the task arguments (`arg-numberOfClusters`); *iii)* the size of the task output (`out-output-size`) measured in number of bytes; *iv)* the information of the used resource (`mach-TotalCpus`, `mach-CpuFrequency`, `mach-JavaMFlops` and `mach-TotalMemory`); and *v)* the execution time of the task (`time`) measured in seconds. Next section details how the performance database are used in the prediction of the performance information.

**Prediction Scheme** The current version of the framework estimates the information using the k-Nearest Neighbors (k-NN) algorithm. Such algorithm selects the most similar instances to a given *query instance* among those that are comprised in a dataset. To do that, the algorithm computes a distance measure of the query instance and the dataset instances. Then retrieves those $k$ instances with the smallest distance to the query instance. An average of the interest attribute (execution time or output size) for each one of the $k$ instances is computed. Note that both the execution time attribute as the output size are of type numeric.

From the analysis of the attributes included on a performance dataset, three different groups can be identified: *task attributes*, *machine attributes* and the *time attribute*. Depending on the type of prediction required by PPSA, such attributes are included or ignored in the prediction process. The attributes selection is performed as follows:

– For the estimation of the size of a task output, the corresponding performance dataset is used but attributes related with other outputs (a task can have more than one output) are ignored such as well as the attributes related with the computational resources. The size of a task's output only depends on the inputs and arguments of such task.
– For the estimation of the execution time of a task, the corresponding performance dataset is used but, this time, the attributes related with the plugin outputs are ignored. This way, the prediction algorithm only considers the information related with the task inputs and arguments, and the characteristics of the resources. All that information have direct relation with the final execution time of the task.

For each type of prediction the corresponding query instance is constructed including the same attributes considered by the k-NN algorithm.

## 4 Myopic Scheduling Algorithm

A simple but well known scheduling algorithm for workflows is the *Myopic Algorithm* [16]. It schedules tasks in a just-in-time manner selecting the most convenient resource each time. The algorithm takes all the ready-to-execute tasks of the workflow and selects the resource that is expected to complete the task earlier. When all tasks have been scheduled the algorithm finishes. Pseudo-code of the algorithm is shown in Algorithm 1. The myopic algorithm has been implemented on a number of production Grid workflow systems such as Condor [17] DAGMan [18,19] and the ASKALON [16,20,21] workflow management systems.

**Algoritmo 1** Myopic scheduling algorithm.

```
while $\exists task\ not\ completed$ do
    $task$ ← get ready task whose parents have been completed
    $resource$ ← get a resource which can complete $task$ at the earliest time
    schedule $task$ on $resource$
endwhile
```

Note that the method for resource selection does respond to any given order and thus it can impact on the performance of the scheduled workflows.

For the purposes of this work, a prototype of the Myopic algorithm was implemented. The developed algorithm selects, each time, the resource that has the best processing capacity. Because, experiments conducted were considering java-based tasks, such processing capacity is represented by the number of MFlops of the resource's Java Virtual Machine. A more detailed explanation about workflow sub-tasks is given in next section.

## 5 Experiments and Results

For studying the enhancement of PPSA in comparison with the Myopic algorithm a set of experiments was designed according to the characteristics of real workflow applications. Both algorithms were used for the scheduling of workflows of different characteristics. Such workflows were generated in a random fashion. Generated workflows are composed by the real predefined executable tasks.

### 5.1 Workflow Generation

For the purpose of the experiments, a set of workflow applications was constructed in a similar way to the one described in a previous work [11]. The details of the workflow generation process are described as follows.

Workflows can be typified based on a tuple $\langle n, \delta \rangle$ where: *i)* $n$, is defined as the number of tasks comprised in the workflow; and *ii)* $\delta$, is the workflow density factor that is directly related with the number of dependencies between tasks. The factor $\delta$ ranges between 0 and 1, and it is a measure of the degree of independence between tasks in a a workflow. If $\delta$ is lower, the tasks in the workflow are highly independent, and if $\delta$ is greater tasks have more dependencies. Workflows with lower values of $\delta$ are often harder to map due to the large number of independent tasks. By knowing the tuple $\langle n, \delta \rangle$, it is possible to calculate the number of dependencies $d$ of a workflow as:

$$d = \delta \times d_{Max} \tag{3}$$

where $d_{Max}$ is the maximum possible number of dependencies of a workflow with $n$ jobs, $d_{Max}$ can be calculated as:

$$d_{Max} = n \times (n-1)/2 \tag{4}$$

To test the performance of the scheduling algorithms, sample workflows were randomly generated with a number of tasks $n$ ranging from 10 to 100 step by 10 and densities of 0.4, 0.6 and 0.8. For each workflow family defined by $\langle n, \delta \rangle$, 10 different sample workflows were generated. The workflows are constructed by tasks of three different types. The output of each workflow task represents data that is transferred to all its successor tasks. The details about the workflow tasks are discussed in the next subsection.

## 5.2 Workflow Tasks

In order to carry out the experiments under more realistic conditions, typical datamining algorithms and operations [15] were selected as tasks for workflow composition. As explained in the previous subsection, the tasks of the generated workflows corresponds to three different types. The three types of tasks are detailed as follows:

- *Data stage-in*: This task in in charge of the retrieval of a dataset from a repository. This tasks receive as argument the name of the required dataset. The task is responsible for transferring the required dataset to the machine where it is executing. Such task is implemented as a single data transfer between two machines.
- *Clustering Task*: This task groups instances present in a dataset. This task receives as input a dataset to clusterize and a parameter indicating the number of desired clusters. The clustering task uses the K-means clustering algorithm included in the Weka library [22].
- *BayesNet Task*: This task is in charge of the construction of a Bayesian model representing the data. The Bayes classification task uses the Bayesian Classifier included in the Weka library [22].

The use of this kind of tasks reduces the gap between the simulation and the real characteristics and requirements of production workflows. Such tasks were defined as part of an ongoing work on the gridification of the XGENE.ORG [23] application for cross-genome cross-organism expression data analysis.

## 5.3 Tasks Performance Data

As explained on subsection 3.2, PEM requires the availability of performance information for each task type. Such information was obtained by executing several instances of the task types detailed in the previous subsection, varying the inputs and parameters for each one. Such tasks were executed over a number of computing resources with different characteristics. The characteristics of the resources used are summarized in table 1. The interconnection bandwidth between resources is 100 Mbps.

Table 1: Characteristics of the Computing resources.

| Resource Type | Total Cpus | Cpu Frequency | Java MFlops | Total Memory |
|---|---|---|---|---|
| Tesla | 1 | 3.0 GHz | 285.45 | 3 GiB |
| Sikus | 2 | 1.87 GHz | 600.04 | 2 GiB |
| Anika | 2 | 2.0 GHz | 379.70 | 2 GiB |

The generated datasets include 186, 90 and 450 execution instances for the tasks *data stage-in*, *clustering* and *classification* respectively.

### 5.4 Results

This subsection presents the results obtained in a preliminary study of the behavior of the two scheduling algorithms previously discussed. The study consists on the scheduling of 300 different workflow applications using both, PPSA and the Myopic algorithm. The set of available resources was composed by 6 machines with similar characteristics of those presented on Figure 1, and an interconnection bandwidth of 100 Mbps was assumed. For comparing the performance of both algorithms, the makespan metric is used. Results are separated on three groups according to the different workflow densities (i.e. values of $\delta$).

In subfigure 4a, the results obtained for workflows with density $\delta = 0.4$ are presented. The figure presents the average makespan for workflow groups with the same number of tasks (i.e. with the same value of $n$). In all the cases, PPSA presents smaller makespans that the Myopic algorithm. The figure also shows the standard deviation of the computed makespans. The makespans obtained by the Myopic algorithm present a large standard deviation in comparison with those obtained by PPSA.

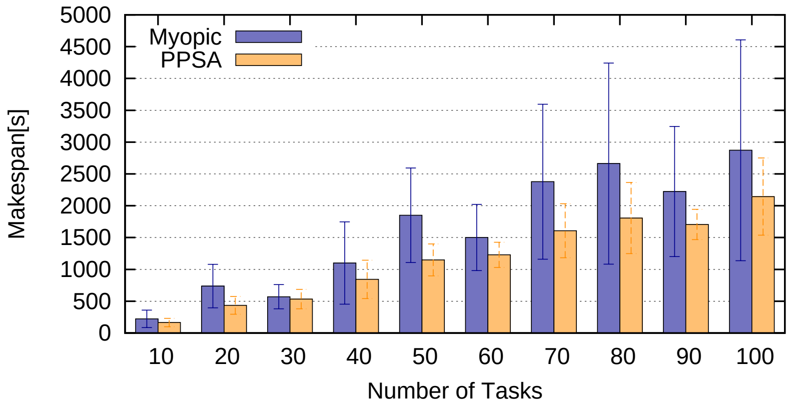
Subfigures 4b and 4c, present the results for workflows with density $\delta = 0.6$ and $\delta = 0.8$ respectively. As in the case of workflows with density $\delta = 0.4$, PPSA overcomes the Myopic algorithm in terms of makespan minimization and presents a small variability in the results obtained.

In general it can be seen that the Myopic algorithm presents high standard deviations on the obtained makespans. These deviations can be explained on the opportunistic behavior Myopic algorithm. A bad selection of resources in earlier tasks may affect the overall makespan of the workflow. In opposition, PPSA presents less variability in the results obtained (i. e. PPSA presents a more deterministic behavior). That is because it considers more resource assignments and thus mitigates the effects of non proper resource selections. Another point to note is related with the mean makespans of the Myopic algorithm. It can be seen that there is not a clear tendency in the mean makespan values obtained by the Myopic algorithm.

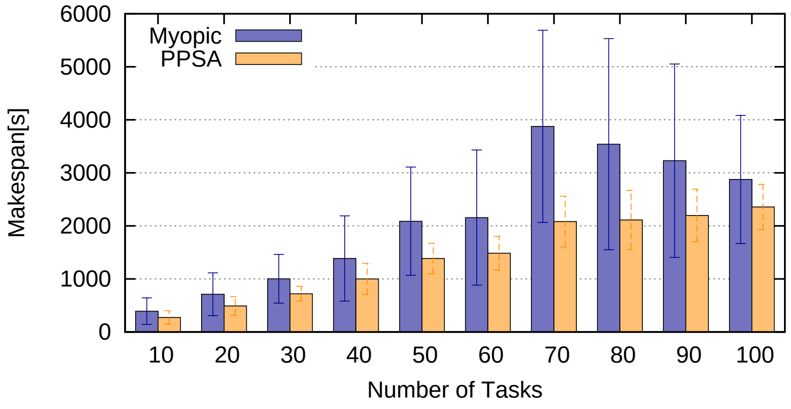
For measuring the degree of improvement of PPSA in comparison with the Myopic algorithm, the speedup measure is used. The makespan speedup for each group of workflows with the same values of $\delta$ and $n$, is calculated as:

$$S_{n,\delta} = \frac{T_{n,\delta}^{Myopic}}{T_{n,\delta}^{PPSA}} \tag{5}$$

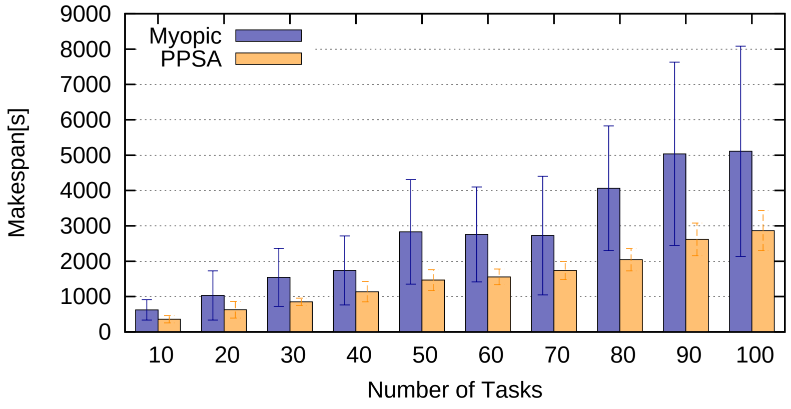
where $T_{n,\delta}^{Myopic}$ is the average scheduling makespan for workflows with $n$ number of tasks and a density of $\delta$, by using the Myopic algorithm. And $T_{n,\delta}^{PPSA}$ is the average scheduling makespan for workflows with $n$ number of jobs and a density of $\delta$, by using PPSA. In figure 5 the speedups for workflow families with densities 0.4, 0.6 and 0.8 ($wf_{\delta=0.4}$, $wf_{\delta=0.6}$ and $wf_{\delta=0.8}$) are presented.



(a) Makespans for workflows with density $\delta = 0.4$.



(b) Makespans for workflows with density $\delta = 0.6$.



(c) Makespans for workflows with density $\delta = 0.8$.

Fig. 4: Makespans obtained for workflows according to the three dependency factors.

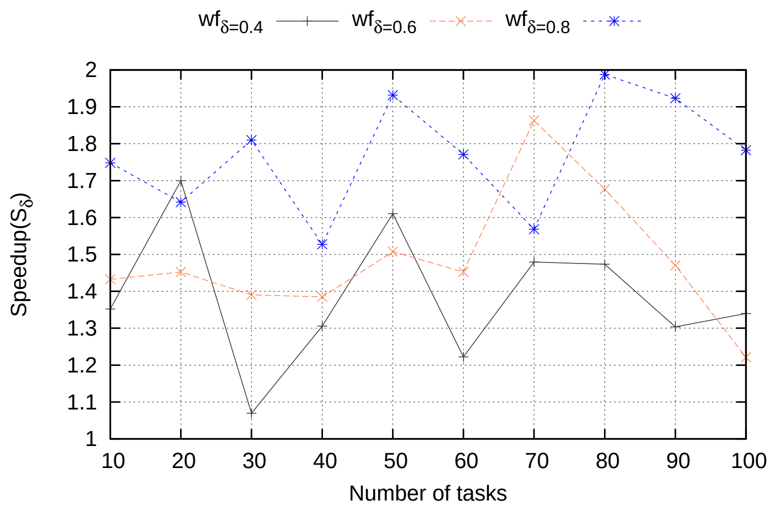


Fig. 5: Speedups obtained for each workflow family.

It can be seen that the speedup curves does not follow any specific pattern. The shape of such curves is because of the high variability of the schedules generated by the Myopic algorithm.

To obtain a representative measure of all workflows with a particular value of $\delta$. The average speedup for $\delta$ is calculated as:

$$S_\delta = avg(S_{n,\delta}) \quad n = 10 \ldots 100 \, step \, by \, 10 \tag{6}$$

The average speedups obtained for the three values of $\delta$ were: $S_{0.4} = 1.38$, $S_{0.6} = 1.48$ and $S_{0.8} = 1.77$. Best speedups were obtained on more dense workflows (largest values of $\delta$). This is because such kind of workflows present a smaller degree of parallelism and thus have critical paths with a biggest number of tasks. And then, delays introduced by the selection of non-proper resources impact on the overall makespan. The speedups can be expressed as percentages of the workflow makespan improvements with the expression: $perc(S) = (1 - 1/S) \times 100\%$. In table 2 a summarization of the speedup values is given.

Table 2: Average speedups and average, maximum and minimum speedup percentages.

| $\delta$ | Speedup($S_\delta$) | $perc(S_\delta)$ | $min\{perc(S_{n,d})\}$ | $max\{perc(S_{n,d})\}$ |
|---|---|---|---|---|
| 0.4 | 1.38 | 26% | 6% | 41% |
| 0.6 | 1.48 | 31% | 17% | 46% |
| 0.8 | 1.77 | 42% | 34% | 49% |

## 6 Concluding Remarks

This paper describes the performance prediction workflow scheduling algorithm (PPSA). PPSA is tightly coupled with Nonparametric Regression (NR)-based performance prediction module. In order to test PPSA, a set of simulation experiments was conducted. Real tasks execution data was used to feed the performance prediction module. A set of workflows composed by three different task types was randomly generated. For measuring the quality of the results of PPSA, the generated workflows were also scheduled with the Myopic algorithm, which is integrated in well known production middlewares such as DAGMan and ASKALON.

Results of a preliminary study showed that PPSA overcomes the quality of the resource mappings in comparison with the Myopic algorithm. The workflow makespans of PPSA schedules are smaller in comparison with the ones generated by the Myopic algorithm. Average makespan reductions of 1.38 were obtained for workflows with density $\delta = 0.4$. For workflows with a density $\delta = 0.6$ the makespans were reduced in a 1.48. Finally, for workflows with a density $\delta = 0.8$ the makespans were reduced in a 1.77. In the best case the makespan reduction was 49% and the worst was 6%. Such range of makespan improvements are produced because of the high variability in the makespan obtained by the Myopic algorithm. The average makespan reductions obtained were 24%, 31% and 39% for workflows with densities $\delta = 0.4$, $\delta = 0.6$ and $\delta = 0.8$, respectively.

Although PPSA presents exponential complexity on the number of workflow tasks, it still produces reduced makespan schedules in an acceptable time without degrading the overall performance.

Currently, we are working on studies to determine which is the proper size of the performance datasets. The performance datasets need to be small enough to allow fast predictions without loosing the accuracy of the estimated information. We are studying mechanisms for the maintenance of the datasets need to be studied.

Additionally, we are working on the development of a framework for distributed data mining (DDMF) and the adaptation of the XGENE.ORG application for its execution on grid environments.

## 7 Acknowledgements

The first author want to thank CONICET, for the PhD fellowship granted. The financial support provided by ANPCyT through PAE-PICT 2312 project. Also, financial support through MINCyT (Argentina) and MEYS (Czech Republic) bilateral cooperation project, identified by codes RC0904 and MEB111005 respectively, is gratefully acknowledged.

## References

1. Jia Yu and Rajkumar Buyya. A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing*, 3:171–200, 2005.

2. Fatos Xhafa and Ajith Abraham. Meta-heuristics for grid scheduling problems. In Fatos Xhafa and Ajith Abraham, editors, *Metaheuristics for Scheduling in Distributed Computing Environments*, volume 146 of *Studies in Computational Intelligence*, pages 1–37. Springer Berlin / Heidelberg, 2008.
3. Jia Yu, Rajkumar Buyya, and Kotagiri Ramamohanarao. Workflow scheduling algorithms for grid computing. In Fatos Xhafa and Ajith Abraham, editors, *Metaheuristics for Scheduling in Distributed Computing Environments*, volume 146 of *Studies in Computational Intelligence*, pages 173–214. Springer Berlin / Heidelberg, 2008.
4. Brian Reistad and David K. Gifford. Static dependent costs for estimating execution time. *SIGPLAN Lisp Pointers*, VII:65–78, July 1994.
5. Jaehyung Yang, Ishfaq Ahmad, and Arif Ghafoor. Estimation of execution times on heterogeneous supercomputer architectures. *Parallel Processing, International Conference on*, 1:219–226, 1993.
6. Hong-Linh Truong, Thomas Fahringer, Georg Madsen, Allen D. Malony, Hans Moritsch, and Sameer Shende. On using scalea for performance analysis of distributed and parallel programs. In *Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM)*, Supercomputing '01, pages 34–34, New York, NY, USA, 2001. ACM.
7. Michael A. Iverson, Füsun Özgüner, and Lee Potter. Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment. *IEEE Trans. Comput.*, 48:1374–1379, December 1999.
8. Rich Wolski, Neil T. Spring, and Jim Hayes. The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Gener. Comput. Syst.*, 15:757–768, October 1999.
9. D. P. Spooner, S.A. Jarvis, J. Cao, S. Saini, and G. R. Nudd. Local grid scheduling techniques using performance prediction. *IEEE PROCEEDINGS COMPUTERS AND DIGITAL TECHNIQUES*, 150(2):87–96, 2003.
10. Ming Wu and Xian-He Sun. Grid harvest service: a performance system of grid computing. *J. Parallel Distrib. Comput.*, 66:1322–1337, October 2006.
11. David Monge and Carlos García Garino. Improving Workflows Execution on DAGMan by a Performance-driven Scheduling Tool. In R. Orozco and A. Fernández, editors, *Proceedings of the Third Symposium on High-Performance Computing (HPC2010) in Latin America, 39 JAIIO*, volume 3, pages 3271–3285, Buenos Aires, Argentina, Aug 2010. SADIO, SADIO.
12. David Monge and Carlos García Garino. A Constraint Optimization based Scheduler for Distributed Computing Workflows. In S. Castro and J. Orozco, editors, *Proceedings of the Second Symposium on High-Performance Computing (HPC2009) in Latin America, 38 JAIIO*, volume 3, pages 159–174, Mar del Plata, Argentina, 2009. SADIO, SADIO.
13. David Monge and Carlos García Garino. Heurísticas novedosas de Constraint Optimization aplicadas al Scheduling de Workflows de Computación Distribuida. In N. Sirsmovitsch, A. Mirasso, and A. P. Arena, editors, *Anales del V Encuentro de Investigadores y Docentes de Ingeniería 2009 - EnIDI 2009*, volume 5, pages 122–136, Los Reyunos, San Rafael, Argentina, Nov 2009. UTN-FRM, UTN-FNSR, UNCuyo-FI, UNCuyo-FCAI.
14. Ewa Deelman, Gaurang Mehta, Gurmeet Singh, Mei-Hui Su, and Karan Vahi. Pegasus: Mapping large-scale workflows to distributed resources. In Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Shields, editors, *Workflows for e-Science*, pages 376–394. Springer London, 2007.

15. Eibe Frank & Mark A. Hall By Ian H. Witten. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufman, 3rd edition edition, Jan 2011.
16. Marek Wieczorek, Radu Prodan, and Thomas Fahringer. Scheduling of scientific workflows in the ASKALON grid environment. *SIGMOD Rec.*, 34:56–62, September 2005.
17. Todd Tannenbaum, Derek Wright, Karen Miller, and Miron Livny. Beowulf cluster computing with windows. chapter Condor: a distributed job scheduler, pages 307–350. MIT Press, Cambridge, MA, USA, 2002.
18. Douglas Thain, Todd Tannenbaum, and Miron Livny. *Condor and the Grid*, pages 299–335. John Wiley & Sons, Ltd, 2003.
19. Peter Couvares, Tevfik Kosar, Alain Roy, Jeff Weber, and Kent Wenger. Workflow management in condor. In Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Shields, editors, *Workflows for e-Science*, pages 357–375. Springer London, 2007.
20. Thomas Fahringer, Alexandru Jugravu, Sabri Pllana, Radu Prodan, Clovis Seragiotto, Jr., and Hong-Linh Truong. ASKALON: a tool set for cluster and Grid computing: Research Articles. *Concurr. Comput. : Pract. Exper.*, 17:143–169, February 2005.
21. Thomas Fahringer, Radu Prodan, Rubing Duan, Jüurgen Hofer, Farrukh Nadeem, Francesco Nerieri, Stefan Podlipnig, Jun Qin, Mumtaz Siddiqui, Hong-Linh Truong, Alex Villazon, and Marek Wieczorek. ASKALON: A Development and Grid Computing Environment for Scientific Workflows. In Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Shields, editors, *Workflows for e-Science*, pages 450–471. Springer London, 2007.
22. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11:10–18, November 2009.
23. Matěj Holec, Jiří Kléma, Filip Železný, Jiří Bělohradský, and Jakub Tolar. Cross-genome knowledge-based expression data fusion. In William Loging, Mukesh Doble, Zhirong Sun, and James Malone, editors, *International Conference on Bioinformatics, Computational Biology, Genomics and Chemoinformatics (BCBGC-09)*, 2009.