Predicting the Execution Time of Grid Workflow Applications through Local Learning*

Farrukh Nadeem[1,2] and Thomas Fahringer[2]

[1]Department of Computer Science, National University of Computer and Emerging Sciences, Lahore, Pakistan

[2]Institute of Computer Science, University of Innsbruck, Austria

farrukh.nadeem@nu.edu.pk, {farrukh,tf}@dps.uibk.ac.at

ABSTRACT

Workflow execution time prediction is widely seen as a key service to understand the performance behavior and support the optimization of Grid workflow applications. In this paper, we present a novel approach for estimating the execution time of workflows based on Local Learning. The workflows are characterized in terms of different attributes describing structural and runtime information about workflow activities, control and data flow dependencies, number of Grid sites, problem size, etc. Our local learning framework is complemented by a dynamic weighing scheme that assigns weights to workflow attributes reflecting their impact on the workflow execution time. Predictions are given through intervals bounded by the minimum and maximum predicted values, which are associated with a confidence value indicating the degree of confidence about the prediction accuracy. Evaluation results for three real world workflows on a real Grid are presented to demonstrate the prediction accuracy and overheads of the proposed method.

1. INTRODUCTION

Grid workflows from scientific and business domains typically consist of several different activities (executables, services, etc.) with complex control flow and data flow dependencies among them. Execution of such workflows in large scale computational Grids, like Grid5000 [25], EGEE [4], etc., is commonly accomplished through a workflow composition and runtime environment like ASKALON [5] for distributed execution of workflow activities. The workflow runtime environment depends on online workflow execution time predictions to guide the performance-oriented optimization of the workflows.

Predicting the execution time of a workflow in the Grid is a complex problem and has been largely ignored so far due to the execution of workflow activities in a distributed fashion, involvement of several Grid resources (multiple Grid sites, LAN/WAN, etc.), external load, dynamic behavior of the Grid, inherent architectural and functional heterogeneity of Grid resources, and different structures of the workflows.

In this paper we introduce a *Local Learning Framework* for workflow execution time prediction, which is based on static information (number of activities in the workflow, control and data flow dependencies among workflow activities, etc.) and dynamic information about the execution of the workflows (through execution traces). This information about workflows is stored in a repository whose data (referred as *workflow data set* or simply *data set*) is used for *local learning* (*LL*). In the course of this paper, we refer to each instance of the data in the *data set* as *data instance*. The workflows are parameterized in terms of *attributes* (determined from the repository) defining workflow static and dynamic information (Section 2.2). The importance of different attributes w.r.t. their impact on execution time of the workflow is determined by attribute weights, which are dynamically determined through an *evolutionary algorithm* (Section 3). These weights are optimized considering the entire data set (to generalize effects of different values of attributes) as well as data subsets (to include effects of specific values of the attributes) and the best weights are selected adaptively (Section 3.2). Our local learning framework (*LLF*) employs hybrid metrics to find similarities in different workflows. The workflows identified to be similar (Section 2.1) are selected for *LL*, and the data set corresponding to the selected workflows is named as *local data set*. One instance of the *local data set* is referred as *local data instance*. We introduce a notion of *distance class* (Section 4) to dynamically select the size of *local data* such that the overall prediction error is minimized. We employ three induction models (Section 5) to predict workflow execution times (called point predictions) from the selected *local data*. A confidence value (ranging between 0 and 100) is associated with each prediction to indicate the degree of confidence about the prediction accuracy. A confidence value 100 means that the prediction is accurate, and a confidence value 0 means that the prediction is unreliable. To indicate possible variations in the predicted execution of a workflow, the minimum and maximum predicted execution times are provided as an interval

*The work described in this paper has been partially funded by the "Higher Eduction Commission" (HEC) of Pakistan, the EC funded edutain@grid project and the project "Parallel Computing with Java for Manycore Computers" funded by the Tiroler Zukunftsstiftung".

SC09 November 14-20, 2009, Portland, Oregon, USA.

(called interval prediction). A prototype of the system based on our approach has been implemented and integrated into ASKALON Grid workflow composition and execution environment.
The contributions in this paper can be summarized as follows: definition of distance classes to dynamically select *local data* for workflow performance predictions; dynamic selection of *local data* to minimize the prediction error; execution time prediction for distributed execution of full workflows on multiple Grid sites; the maximum and the minimum execution time prediction as prediction interval, to indicate possible variations in the prediction; measurement of *prediction confidence* reflecting the degree of confidence in the predicted values; identification of important attributes describing the workflow performance on a Grid.

1.1 Executing Scientific Workflows in the Grid

A Grid workflow, commonly represented by a *directed acyclic graph* (DAG), can be seen as a collection of computational tasks (activities) that are processed in a well defined order to accomplish a specific goal [18]. Several Grid environments [3, 22, 15, 5] have been built to effectively support workflows on the Grid. The execution of such workflows is accomplished through a Grid workflow composition and runtime environment (ASKALON in our case) which optimizes executions of the workflow activities and communications during the workflow execution.
Workflow activities may be executed sequentially or in parallel depending upon the workflow structure and optimization capabilities of the Grid runtime environment. In case of parallel activities there may be some delays due to waiting for all of the parallel activities to finish. After submission of an activity to a Grid site, the execution of the activity is mainly determined by local scheduling policies, queue wait overheads, external load and the computational power of the Grid site. The execution of a workflow is finished when the execution of the last activity of the workflow is finished and the output of the workflow is transferred to the site from where a user submitted the workflow. The control flow of workflow activities, and enumeration of loops in the workflow are usually driven by the input problem size. All of these steps determine the overall execution time of the workflow and collectively make the prediction of its execution time a challenging problem.

2. LOCAL LEARNING FRAMEWORK

Local learning methods train a given data set for learning, and find local data to answer a particular prediction query [1]. Our *LLF* consists of two basic components: a *similarity function* to measure similarity (Section 2.1) among different workflows (which involves a distance function) to identify the local data, and an induction model (Section 5) to provide predictions based on the local data set. We employ *LL* for our workflow execution time prediction, because it is generic and flexible to incorporate attributes of various types (numeric/nominal/vector, etc). Furthermore, *LL* can be easily adapted to the situations where some of the workflow attributes are missing. Moreover, attributes can be interpreted with different importance in the distance function by assigning different weights to them, which enables us to tune the similarity measure for finding similar workflows.
A main problem to apply *LLF* to a given problem is how to represent the workflows that can be used to support efficient similarity measurements. We represent each workflow as a data point (x, y), where x is a workflow represented through a vector of n workflow attributes $< \alpha_1, \alpha_2, ..., \alpha_n >$ (defined in Section 2.2), α_i represents the i^{th} attribute, and y is the execution time of the workflow represented by x. Each data point can be considered as a point in an n-dimensional *Euclidean Space*, and the similarity between different workflows can be measured using a distance function (Section 2.1), where nearby workflows have higher similarity.

2.1 Similarity Measures

We define a workflow by a set of attributes defining the workflow structure, workflow activities, set of Grid sites, execution environment, local job scheduling policies on the Grid sites, and data transfers. We use the distance between two workflows as a measure of similarity between them. The smaller the distance the higher the similarity of the workflows. The distance D between two workflows i and j is calculated as:

$$D(i,j) = \sum_{r=1}^{n} w_r \times d(\alpha_r(i), \alpha_r(j)). \quad (1)$$

Here w_r represents the weight of an attribute α_r. Weighing of the attributes is addressed in Section 3. $d(\alpha_r(i), \alpha_r(j))$ represents the distance between the r^{th} attribute of workflows i $(\alpha_r(i))$ and j $(\alpha_r(j))$, and is given by adapting the definitions from the Heterogeneous Euclidean-Overlap Metric (HEOM) [27]. The attribute values of the workflows are obtained from the repository. For example, attribute values of the attribute *queue name* may be *production-queue*, *research-queue*, *student-queue* etc. The adapted distance definitions give distance between two values v_1 and v_2 of an attribute α as follows:

$$d(v_1, v_2) = \begin{cases} d\_num_\alpha(v_1, v_2) & \text{if the type of } \alpha \text{ is numeric;} \\ d\_nom(v_1, v_2) & \text{if the type of } \alpha \text{ is nominal;} \\ d\_vec_\alpha(v_1, v_2) & \text{if the type } \alpha \text{ is vector.} \end{cases}$$

The functions $d\_num_\alpha(v_1, v_2), d\_nom(v_1, v_2)$, and $d\_vec_\alpha(v_1, v_2)$ are defined by Equations (2), (3) and (4), respectively.

$$d\_num_\alpha(v_1, v_2) = \frac{|v_1 - v_2|}{max_\alpha - min_\alpha} \quad (2)$$

$$d\_nom(v_1, v_2) = \begin{cases} 0 & \text{if } v_1 = v_2 \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

where max_α and min_α are the maximum and minimum values for attribute α, observed in the workflow data set.

$$d\_vec_\alpha(v_1, v_2) = \frac{\sum_{i=1}^{n_\alpha} d(v_1(i), v_2(i))}{n_\alpha} \quad (4)$$

where n_α represents the total number of attributes in an attribute vector α; $v_1(i)$ and $v_2(i)$ represent values of the i^{th} attribute in the two vectors v_1 and v_2, respectively.
Different numeric attributes are measured on different scales, and if these differences are used directly the effects of some of the attributes might be completely dwarfed by others that have larger scales of measurements [28]. To eliminate this effect, we normalize all numeric attribute values to ensure a value between 0 and 1 by dividing them with their range $(max_\alpha - min_\alpha)$ as in Equation (2). For nominal attributes,

the difference between two values that are not same, is taken to be one, otherwise zero. Thus no scaling is required in this case. This policy of normalizing distances of heterogenous attributes to a uniform scale (between 0 and 1) enables us to add them in the distance measurements.
We ensure the maximum distance between two attribute values when at least one of them is missing. For nominal attributes the maximum distance is taken as 1. In case of numeric attributes, if one of them is missing, the maximum distance is calculated as one minus the normalized value of the available attribute, otherwise 1.

2.2 Similarity Attributes

The similarity among different workflows is defined based on the following workflow attributes.

2.2.1 Workflow Activities and Workflow Structure Attributes

Atomic workflow activities are characterized on the basis of:
- *Executables*: the names of the executables (exe) used for each activity a, represented as $< a, exe >$.
- *Activity Type*: an activity is executed as batch, serial, or interactive job.
- *Problem size*: is input data of an activity, specified by the user while submitting the workflow, e.g. through command line parameters. This is one of the attributes that have large impact on workflow execution time.
Wombacher et al. [29] investigated several attributes to describe a workflow structure. As proposed by [29], we consider workflow structure similarity in term of:
- *Workflow name*: name of the workflow submitted by the user. Different users usually store their customized workflows with different names. Workflows with the same names are likely to have similar structures and thus have high probability of similar executions.
- *Workflow activities*: correspond to computational tasks that are executed by *task-executables* or web services. Workflow activities are represented as a set of activity names $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$. The order of activities in $\mathcal{A}$ represents the order of activities in the workflow. *if* and *loop* conditions in the workflow are represented as $< if, a_{true}, a_{false} >$ and $< loopType, a_1, \ldots, a_n >$, respectively. Here a_{true} and a_{false} denote activity names in case when the *if* condition is true and false, respectively. *loopType* describes the type of the loop (sequential/parallel) and $a_1, \ldots, a_n$ represent the n activities inside of a loop. The data flow (indicating production and consumption of data) between the activity producing data a_{src} and the activity consuming that data a_{sink}, is represented as $< a_{src}, a_{sink} >$.

2.2.2 Grid Sites and Grid Site State Attributes

We consider the following attributes to describe similarity of the Grid sites.
- *Set of Grid sites*: considered by the runtime environment to map workflow activities. The workflow execution time varies for different Grid sites. Each Grid site s is represented by a vector $< s, c_{req} >$ where c_{req} represents the number of CPUs requested for the execution of a job on s.
- *Single Grid site*: we consider single Grid sites in terms of CPU architecture (CPU_{arc}), speed (CPU_{spd}), and memory (mem) as $< CPU_{arc}, CPU_{spd}, mem >$.
Grid sites with the same states (running jobs, jobs in the queue, and parallel running jobs) have a higher probability of similar throughput. The state of a Grid site s is defined for every activity which is submitted for execution or executing on s by the queue wait overheads and the external load. We define the queue wait overheads on a Grid site in terms of the following attributes.
- *Running jobs*: Grid site state for other jobs running at the time when the job of an activity a is submitted to a Grid site, is represented by a vector of number of jobs currently running j_{run}, their occupied number of CPUS j_c, and min. (t_{min}) and max. time (t_{max}) required by running jobs as $< j_{run}, j_c, t_{min}, t_{max} >$.
- *jobs in the queue*: Grid site state for other jobs in the queue at the time of submission of activity a is represented by vector of number of other jobs already in the queue (to which the activity job was submitted) or other higher priority queues (if any) j_{queue}, required number of CPUs c_{queue}, and total required time t_{queue} by j_{queue} as $< j_{queue}, c_{queue}, t_{queue} >$.
The external load due to other jobs running concurrently with activity a on the same Grid site, is defined in term of the following attribute.
- *Parallel running jobs*: are represented by a vector of jobs j_{par} running concurrently with activity a and their occupied number of CPUs c_{par} as $< j_{par}, c_{par} >$.
Collectively, state of a Grid site s for a submitted activity a is represented as a vector $<a, s, <j_{run}, j_c, t_{min}, t_{max}>, <j_{queue}, c_{queue}, t_{queue}>, <j_{par}, c_{par}> >$.

2.2.3 Execution Priorities and Local Scheduling Policy Attributes

It is a common practice in the Grid to implement different execution priorities based on virtual organizations (VO). Grid sites may have different queues (such as *production-queue*, *research-queue*, *student-queue*) which imply different priorities (e.g. defining access limits [11]).
The local scheduling policy of a Grid site has a large impact on the execution time of the workflows where Grid sites have high usage rates [6]. The following attributes are considered to describe the execution priorities and local scheduling policy information.
- *User name*: name of the user executing a workflow.
- *VO name*: to which the user belongs.
- *Queue name*: where the job of an activity was submitted.
- *Job submission system*: name of the local job submission system on a Grid site, (e.g. LSF [17], SGE [21], etc).
- *Local scheduling policy*: name of the local scheduling policy on each Grid site, i.e. first in first out, backfill, etc.

2.2.4 Execution Environment Attributes

The similarity of the execution environment is considered in terms of:
- *Grid middleware*: executes a workflow on behalf of a user, e.g. ASKALON. Different middlewares use different high level services for management of workflow activities and thus imply different workflow execution times.
- *Scheduling algorithm*: used to map workflow activities on different Grid sites. This is also one of the attributes that have high impact on the workflow execution time.
- *Time of submission*: workflows submitted during the same time period (peak or off-peak time) have higher probability of similar execution environment.
- *Maximum execution time*: provided by the user, is among the most important attributes to measure workflow similar-

ity when the problem size is not explicitly mentioned by the user.

2.2.5 Data Transfer Attributes

Distributed execution of workflow activities commonly involves data transfers among activities. The size of the data to be transferred is usually a function of input problem size, which has already been considered in Section 2.2.1. In order to measure similarity of each data transfer during the workflow, the following network attributes are considered.
- *Network bandwidth*: at the time of the data transfer.
- *Network latency*: at the time of the data transfer.
- *Number of parallel transfers*: used to transfer the data.

The information corresponding to all these workflow attributes is obtained collectively from ASKALON repository and log files of local resource managers on individual Grid sites through an automated tool.

3. WORKFLOW ATTRIBUTE WEIGHING

The workflow attributes have different impacts on workflow execution time. This raises the need to emphasize the attributes in the order of magnitude of their impacts on workflow execution time for better similarity measurements. We achieve this goal by assigning different weights to the workflow attributes (see Equation 1). We design an *evolutionary algorithm* (Section 3.1) to dynamically optimize these weights. Weights are determined based on the entire data set (referred as *Global Weighing*) as well as the data sets divided based on a prediction query-specific pivot attribute value (referred as *Local Weighing*). A prediction query consists of user specified workflow attribute values for which workflow execution time prediction is required. For example, a prediction query may be to estimate execution time of workflow *Wien2K* [16] on three Grid sites $ASG1$, $ASG2$, and $ASG3$ using HEFT [9] scheduling algorithm. The selection of either global or local weights is made adaptively based on generalization of prediction error and bias-variance analysis of the predictions (see Section 3.2.1). The generation, optimization and adaptive selection of the attribute weights are addressed in the following sections.

3.1 Dynamic Weighing through Evolutionary Algorithm

An evolutionary algorithm (EA) is a population based metaheuristic optimization technique inspired by process biological evolution: variation, reproduction and selection [2]. EA allows more flexible and efficient representation, more sophisticated operators and higher efficiency than classical approaches for evolution like Genetic Algorithms (GA's). Individuals in the population play a role of candidate solutions (attribute weights in our case), and are represented by high level data structures.

The evolution process starts with generation of the *initial population* (selected at random), where each individual in the population is a set of weights $\{w_i\}$ for the set of workflow attributes $\{\alpha_i\}$. These individuals evolve for optimal values over different *evolution generations*. The *evolution landscape* (set of possible values during the evolution process) of individuals consists of all the natural numbers below a given threshold. The process of evolution in each generation consists of evaluating the fitness of the individuals in the population, selecting which individuals will be operated on to produce the new generation, and operating on the selected individuals to obtain the next generation. This process is repeated until the convergence criteria is satisfied (usually a required fitness of individuals or number of generations). Two special operators of $roleOver$ and $crossOver$ are designed to perform evolutionary operations over population individuals. The $roleOver(val, n)$ operator adds $n \mid n \in \mathbb{Z}$ to an existing value val of an individual, and maintains its overall value below the given threshold. The $crossOver(\alpha, x_1, x_2)$ operator swaps the attributed weights in the two individuals (x_1, x_2) by selecting α as a pivot value, to form two new individuals. These operators are applied to produce the new generation in the same fashion as native operators in GA's. The individuals for evolution to the next generation are selected through *elitism* and *Stochastic Universal Sampling* (SUS) [2]. Under the policy of elitism, the best $\lceil 10\% \rceil$ individuals from a population are transferred unmated [2] to the next generation. The SUS is implemented through a *roulette wheel* with n pointers [2], where n is number of individuals to be selected through SUS. The space on the roulette wheel is assigned to the individuals in proportion to their fitness given by Equation 5.

For the execution time prediction of the workflows, the goal of our evolutionary algorithm is to minimize the cross validation prediction error and maintain a minimum range of fitnesses among the individuals. To accomplish this goal, we consider the locally weighted average of the squared cross validation prediction error [1] as a fitness function, which is defined as follows:

$$fitness = \frac{\sum e_i K(d(x_i, q))}{\sum K(d(x_i, q))} \tag{5}$$

where e_i represents the squared leave-one-out cross validation prediction error [1], K represents the kernel function (see Section 5), $d(x_i, q)$ represents the distance between a workflow x_i and given prediction query q.

To ensure effectiveness and efficiency of our approach, we constrain the maximum size of the population to the number of attributes describing the workflow.

3.2 Adaptive Selection of Weights

In this section, we describe how to adaptively select workflow weights based on a global or local weighing scheme. The global weighing scheme aims at generalization of performance effects over the entire data set, whereas the local weighing scheme targets to focus on the performance effects specific to subsets of the data set divided based on the prediction query specific attribute values. We consider the meta scheduling algorithm, the number of Grid sites, and the workflow name as pivot attributes to divide the data set for local weighing. While global weights can be used to achieve a better generalization of performance effects, the *variance* in predictions for certain specific queries may be high. Similarly, while the use of local weights is better suited to model the performance effects specific to a given prediction query, the *bias* (due to suffering from overfitting in the predictions) may be high. To escape from this dilemma, we make a bias-variance tradeoff for our predictions.

3.2.1 Bias-Variance Tradeoff

Let $T = \{(x_1, y_1), ..., (x_n, y_n)\}$ represents the data set, and $F(q, T)$ is a function that determines the prediction for query q over T. According to Haykin [10] the mean squared error $MSE(q, T)$ of prediction can be decomposed as:

$$MSE(q,T) = \mathbb{E}_T[(\mathbb{E}[y|x] - F(q,T))^2]$$
$$= \mathbb{E}_T[(F(q,T) - \mathbb{E}_T[F(q,T)])^2] + (\mathbb{E}_T[F(q,T)] - \mathbb{E}(y|x))^2$$

Here $\mathbb{E}_T$ represents the expected value calculated over T. For details the reader may refer to [10]. This first component in the right part of this equation is the variance and the second one is the bias. This indicates that a model with less bias (tuned globally) may still have high variance. Likewise a highly biased model may exhibit poor accuracy if its variance is high. Thus, a tradeoff has to be made between bias and variance to optimize the average prediction accuracy. The selection of either global or local weights is made based on this tradeoff.

The algorithm for adaptive selection of global or local weights is presented in Algorithm 1. The algorithm starts with the extraction of the local data sets for the given pivot attribute values (line 3). In the next step, global and local weights are determined through the evolutionary algorithm (lines 5 to 8). Further, the algorithm compares all sets of local weights with global weights, based on number of data instances (n), general prediction error (ne) and bias-variance tradeoff. The ne is the average leave-one-out prediction error normalized by the measured (real) execution time. At the beginning, we try to generalize weights for the entire data set, by selecting the global weights as the best. For a set of local weights to supersede the selected best weights, we design the following conditions. First, n must be larger than a given threshold to hold enough information about workflow (line 10). Second, the bias introduced by local weights must be less than the bias introduced by the current best weights which is enforced through a given threshold of δ_{Bias} (line 11). Third, the variance introduced by candidate weights Var_{ω_i} must be less than or equal to that of the best weights Var_{Ψ} (line 12). In the case when the second and third condition are not met, then prediction error caused by the candidate weights ne_{ω_i} must be less than the prediction error of the current best weights by a given threshold δ_{error} (line 16).

Algorithm 1 aims at combining advantages of both local and global weights. The best weights are employed to find the distance (Equation (1)) between workflows.

Algorithm 1 Dynamic optimization and selection of workflow attribute weights.

Input: T : Training data set
$A = \{\alpha_1, ..., \alpha_n\}$ is the set of pivot attribute values;
Threshold values of data size δ_n, bias δ_{Bias} and prediction error δ_{error}
Output: Ω: Set of selected weights

```
1: L, W set to empty sets.
2: for all α_i ∈ A do
3:    L ⇐ L ⋃ getLocalDataSet(T, α_i)
4: end for
5: Ψ ⇐ getEvolWeights(T)
6: for all l_i ∈ L do
7:    W ⇐ W ⋃ getEvolWeights(l_i)
8: end for
9: for all ω_i ∈ W do
10:   if |l_i| > δ_n then
11:      if Bias_ωi * δ_Bias ≤ Bias_Ψ then
12:         if Var_ωi ≤ Var_Ψ then
13:            Ψ ⇐ ω_i
14:         end if
15:      else
16:         if ne_ωi ≤ ne_Ψ + δ_error then
17:            Ψ ⇐ ω_i
18:         end if
19:      end if
20:   end if
21: end for
22: return Ψ
```

4. DISTANCE CLASSES AND DYNAMIC SELECTION OF LOCAL DATA

The similar workflows (referred as *the nearest neighbours* (NNs)) are identified based the distance between workflow data instances and a given prediction query. Instead of selecting a fixed number of NNs, as done by other approaches employing similar frameworks like in [11], we dynamically select the number of NNs. For this purpose, we define *distance class*, and use it for dynamic selection of NNs. A distance class is a set of the data instances equidistant from the given query. One distance class is created for each unique distance between the data instances and the given query. The dynamic selection of a number of NNs is made by iterative evaluation of workflow data instances from different distance classes (starting with minimum distance and including the next larger distance in the next iteration) such that the *cross validation prediction error* [1] from selected NNs, computed as $\sum_{i=1}^{l}(p - m_i)^2$, is minimized. Here, l is the total number of NNs, p is the predicted execution time from the selected NNs, and m_i is the measured execution time of the ith NN. The notion of distance class and dynamic selection of distance classes enable us to achieve higher accuracy by converging to a global minima of all the data instances that are at equal distance from the query point. In contrast, selecting a fixed number of NNs may converge to the local minima.

5. INDUCTION MODELS FOR WORKFLOW EXECUTION TIME PREDICTION

After dynamic selection of a number of NNs through distance classes, an induction model is applied to generate the workflow execution time predictions from the selected NNs. The three induction models considered in our current study include *k-Nearest Neighbour*($k-NN$), $k-Weighted\ Average$ and *Locally Weighted Linear Regression*. The $k-NN$ local model simply takes the average of the k closest data points as a prediction of workflow execution time. The second local model *k-Weighted Average* predicts the workflow execution time by averaging the execution times of the k NNs inversely weighted by their distance to the query point. It is defined as:

$$P(q) = \frac{\sum_{j=1}^{k} w_j y_j}{\sum_{j=1}^{k} w_j},$$

where $P(q)$ is the predicted workflow execution time based a given prediction query q, y_j is the execution time of the jth NN, w_j is the weight for the jth NN determined as $w_j = K(D(q, x_i))$. Here $K(d)$ is a kernel function used to determine the weight for the jth NN as a function of its distance d. $K(d)$ is defined as $K(d) = b/d$, where b is the kernel bandwidth (sometimes referred to as the smoothing bandwidth). We set the kernel bandwidth to the distance of the kth nearest neighbour. The third local model *Locally Weighted Linear Regression* predicts the workflow execution time by fitting a linear surface to the selected NNs by using a distance weighted regression [1].

6. EXPERIMENTS

6.1 Scientific Workflows

In current study, we experimentally evaluated our approach for three real world scientific workflows: Wien2k [16], Invmod [26], and MeteoAG [19], when executed through ASKALON on the Austrian Grid [24] testbed.

Wien2k (Figure 1(a)) is a program package for performing electronic structure calculations of solids using density functional theory based on the full-potential augmented plane-wave and local orbital methods. Invmod (Figure 1(b)) aims at the *Water Flow and Balance Simulation Model*, to help in studying the effects of climatic changes on the water balance in order to obtain improved discharge estimates for extreme floods. The MeteoAG (Figure 1(c)) is a workflow for meteorological simulations, which produce atmospheric fields of heavy precipitation cases over the western part of Austria to forecast alpine watersheds and thunderstorms.

Table 1 describes the mean, the minimum and the maximum execution times of each workflow executed on the Austrian Grid and their corresponding execution periods.

A summary of Austrian Grid sites considered for our experiments and the number of activities executed on them are described in Table 2. Most of the Grid sites have different architectures, local resource managers implementing different scheduling policies, multiple submission queues with different priorities for different users.

6.2 Prediction Evaluation

Each data instance in our workflow data set consists of different workflow attribute values and the corresponding workflow execution time. For our experiments, the workflow data set (summarized in Table 1) was divided into two disjoint sets, the training data set (used for LL), and the test data set (used to predict the workflow execution time based on LL from the training data set).

The workflow execution times is predicted as a single estimated value (referred as point prediction). To indicate the degree of confidence about the prediction accuracy, a prediction confidence is associated with each prediction (ranging from 0 (the worst case) and 100 (the best case)). The prediction confidence is provided with a given threshold of variation ($\pm threshold\ value\%$) of the predicted execution time. For example, for a predicted execution time t, a prediction confidence of 95% for a threshold value of 20% means that the client can be confident 95% that the predicted time will be $t \pm 20\%$. The confidence of prediction increase with the increase in the threshold value for the confidence. For the presented experiments, the prediction confidence is measured for a threshold value of 10%. To express possible variations in the predicted execution time, the minimum and maximum execution times of the workflow (t_{min} and t_{max}, respectively) are also predicted (called interval prediction).

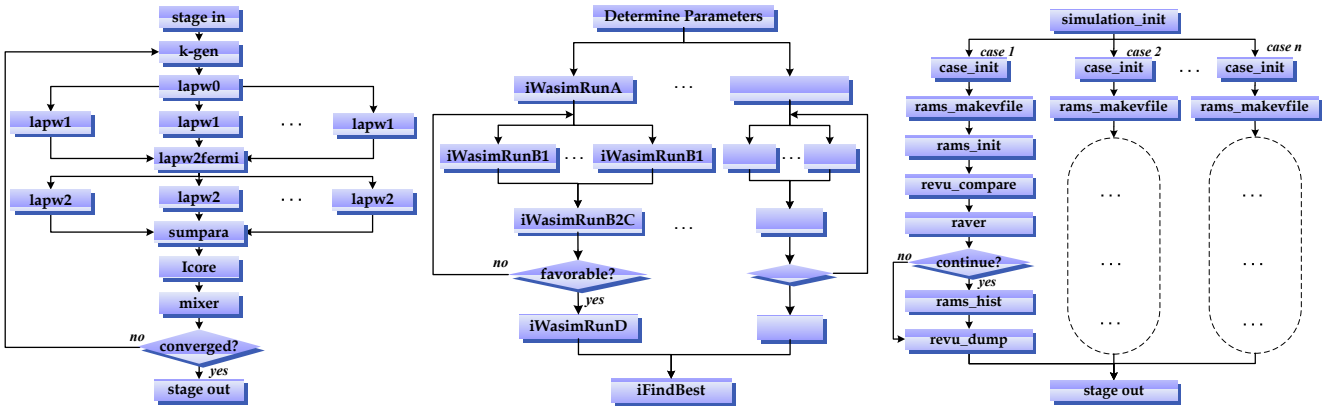
Two kinds of metrics are used to evaluate the effectiveness of our prediction method. First, the prediction accuracy is measured as the *average absolute error* of n predictions, defined as $\sum_{i=1}^{n} |t^i_{pred} - t^i_{real}|/n$. The t^i_{pred} and t^i_{real} denote the ith predicted and measured execution times, respectively. Second, the *normalized prediction error* given by $(t_{pred} - t_{real})/t_{real}$, and the *normalized absolute error* by $|t_{pred} - t_{real}|/t_{real}$, are measured for the comparison of prediction accuracies for the workflows across different Grid sites. The accuracy of the interval predictions is measured by the *interval prediction relative error* measured as $(t_{min} - t_{real})/t_{min}$ if $t_{real} < t_{min}$, or $(t_{real} - t_{max})/t_{max}$ if $t_{real} > t_{max}$.

The accuracy of the predictions is evaluated across different problem sizes of all three workflows and also for different number of Grid sites. Throughout our evaluations, each experiment of measured and predicted execution times was repeated 10 times to eliminate possible anomalies during the experimental process, and their average values are presented in this paper.

Figure 2 shows the maximum and the minimum execution time predictions normalized with the predicted execution time, the prediction confidence and the normalized prediction error for a randomly selected test data set of 280 instances of the MeteoAG workflow. The normalized values of the maximum and the minimum execution times are measured as: $\frac{Maximum\ execution\ time}{predicted\ execution\ time}$ and $\frac{Minimum\ execution\ time}{predicted\ execution\ time}$, respectively. Figure 2 confirms that the predicted execution times are close to the measured execution times and the range of interval predictions (*maximum execution time prediction* − *minimum execution time prediction*) is very small. At the same time the corresponding prediction confidence is high. The median confidence of the execution time predictions for MeteoAG workflow is 100% (mean = 91%), and the normalized absolute error for point predictions (on average) is 9%. The predicted intervals are on average within −10% to +8% of the predicted execution times. For 90% of the predictions (252 out of total 280 predictions) the measured execution time are within our predicted interval. For the remaining 10% of predictions, the prediction interval relative error was 11% on average. The prediction confidence, on average for all 280 predictions is 94%. The selected number of distance classes in these predictions ranged from 1 to 14. The NN count (see Section 4) in the selected distance classes ranged from 1 to 97. The distribution of normalized errors in point predictions from our experiments for MeteoAG workflow is shown in Figure 5(left).

Figure 3 shows the maximum and the minimum execution time predictions normalized with the predicted execution time, the prediction confidence, and normalized prediction error for a randomly selected test data set of 280 instances of the Wien2K workflow. Like for the predictions of the MeteoAG workflow, this figure confirms that the predicted execution times are close to the measured execution times and range of prediction interval is also small, while the prediction confidence is high. The normalized absolute error for point predictions (on average) is 15% where the median predictions confidence is 100% (mean = 92%). The distribution of normalized errors of point predictions across all of our experiments for the Wien2K workflow is shown in Figure 5(right).

The accuracy of point predictions for MeteoAG is higher than that of Wien2K, whereas the accuracy of the interval predictions is higher for Wien2K. For both workflows, we observed an inverse relationship (see Figure 4) between prediction confidence and normalized prediction error, i.e. the normalized prediction error was smaller for the predictions with higher prediction confidence and vice versa. Furthermore, the range of prediction interval also varied inversely to the prediction confidence. Figure 5 confirms that more than 60% of the predictions have a normalized error of 10% or less.



(a) Wien2K Workflow. (b) Invmod Workflow. (c) MeteoAG Workflow.

Figure 1: Structure, activities and dependencies of the three scientific workflows.

Table 1: Characteristics of Wien2K, Invmod and MeteoAG Workflows Executions on the Austrian Grid

Workflow Name	Activity Count	When			Execution Time (minutes)		
		From	To	Duration	Mean	Maximum	Minimum
Wien2K	279516	April 6,2005	June 26, 2007	28 months	10.22	124.26	1.40
Invmod	57646	April 11,2005	Aug 17, 2006	18 months	6.55	91.71	1.35
MeteoAG	101762	May 8,2006	Dec. 10, 2007	20 months	19.59	123.88	1.23

Table 2: Austrian Grid testbed and number of workflow activities executed on different Grid sites.

Grid site*	Location	Architecture	OS	LRM	CPUs	Workflow activities
AGS-1	Innsbruck	NOW, Ethernet Pentium 4, 1.8	Linux	PBS	20	8963
AGS-2	Innsbruck	NOW, Ethernet Pentium 4, 1.6	Linux	PBS	20	24093
AGS-3	Linz	NOW Ethernet, AMD Athelon, 1.6	Linux	Torque	16	13434
AGS-4	Innsbruck	ccNUMA, SGI Altix 350 Itanium 2, 1.6	Linux	Torque	16	85188
AGS-5	Salzburg	ccNUMA, SGI Altix 350 Itanium 2, 1.8	Linux	Fork	16	17021
AGS-6	Innsbruck	NOW Ethernet, Intel Xeon 2.0	Linux	Torque	12	3596
AGS-7	Innsbruck	COW, AMD Opteron 848 dual-core , 2.4	Linux	SGE	80	26294
AGS-8	Linz	NUMA, SGI Altix 3000 Itanium 2, 1.6	Linux	PBS	64	48780
AGS-9	Innsbruck	COW, AMD Opteron 248, dual-core 2.2	Linux	SGE	198	12988
AGS-10	Innsbruck	NOW Ethernet, AMD Opteron 244, 1.8	Linux	Torque	6	1452

*We make the Grid sites names anonymous here.

The next experiment demonstrates how the prediction accuracy changes when problem size and number of Grid sites are varied. We evaluated changes in the accuracy of our predictions for three different problem sizes for MeteoAG, Wien2K and Invmod as shown by Figures 6. A higher problem size implies higher execution times for the workflows. For MeteoAG, the normalized absolute error varied between 0.07 (minimum) and 0.25 (maximum), and the mean normalized absolute error is 0.09. For Wien2K, the normalized absolute error varied between 0.09 (minimum) and 0.23 (maximum), and the mean normalized absolute error is 0.15. In sequel, for Invmod, the normalized absolute error varied between 0.13 (minimum) and 0.21 (maximum), and the mean normalized absolute error is 0.15. We did not notice any increasing/decreasing patterns of the normalized prediction errors by increasing/decreasing problem sizes. Similarly, no patterns of the normalized prediction errors were observed when number of Grid sites were varied (see Figure 6).

To evaluate the effectiveness of our two dynamic weighing schemes (*global weighing scheme* versus *local weighing scheme*), we compared the two schemes against their prediction accuracies for the MeteoAG, Wien2K and Invmod. Figure 7 depicts the normalized absolute error for the three workflows using global and local weighing schemes, for a number of Grid site ranging from 1 to 9. The global weighing scheme better weighed the workflow attributes for MeteoAG and Invmod, while the local weighing scheme appeared to be better for Wien2K. The prediction accuracy by using the global weighing scheme is 17% and 11% higher than the prediction accuracy determined by using the local weighing scheme, for MeteoAG and Invmod, respectively. However, the local weighing scheme outperformed the global weighing scheme by 24% for Wien2K. We found that the selection of the global weighing scheme for MeteoAG was due to a higher bias in local weights (bias-variance tradeoff Section 3.2.1), and in case of Invmod this selection was due to a smaller number of data instances used for local weighing. The selection of local weights for Wien2K was made due to higher variance in predictions from global weights.

For the next experiment, we compared the prediction accuracy from the proposed approach based on local learning as introduced in this paper, with the accuracy of predictions from our previously published method based on *similarity templates* [13]. Figure 8 compares the prediction accuracy of the two approaches for the three workflows, for Grid sites ranging from 1 to 9. It can be seen that the *LLF* delivered

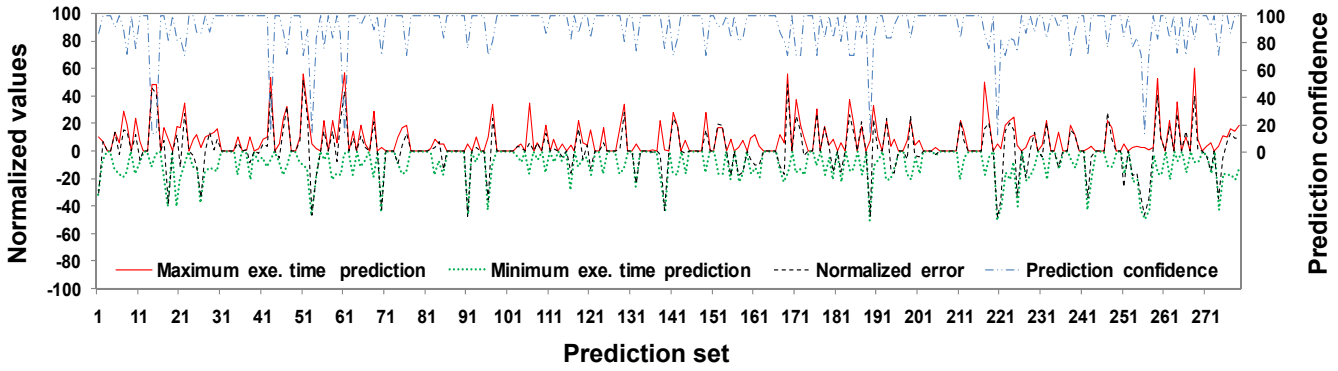


Figure 2: Normalized maximum and minimum execution times predictions, prediction confidence and normalized error of predictions for MeteoAG.

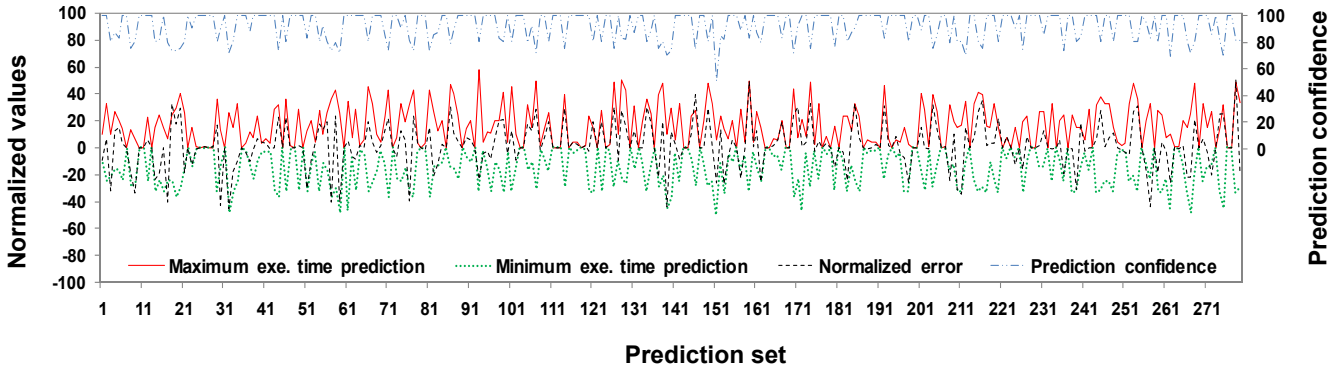


Figure 3: Normalized maximum and minimum execution times predictions, prediction confidence and normalized error of predictions for Wien2K.

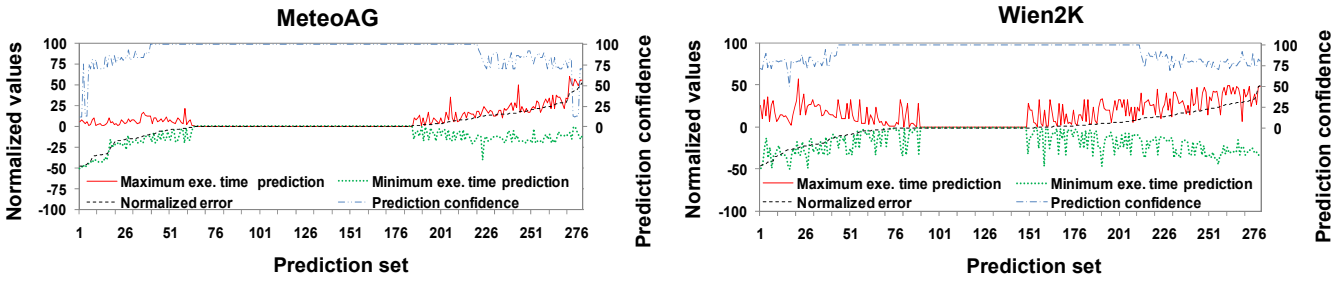


Figure 4: Relationship between prediction confidence, interval prediction and normalized error for MeteoAG and Wien2K.

better accuracy than the similarity templates for all three workflows. The prediction accuracy of the *LLF* outperforms the results of the method based on similarity templates by 22%, 30% and 49% for Invmod, MeteoAG and Wien2K, respectively. Based on these results, we infer that a well designed *LLF* is likely to achieve better prediction accuracy than our previous work on similarity templates. Moreover, for the three workflows, we found (see Figure 9) that the difference in the prediction accuracy of the two methods is directly proportional to the quantity of the training data. We believe that this improvement in the prediction accuracy is due to the large dimensional space of the workflow attributes. Based on our experiments, we observed that *Grid*

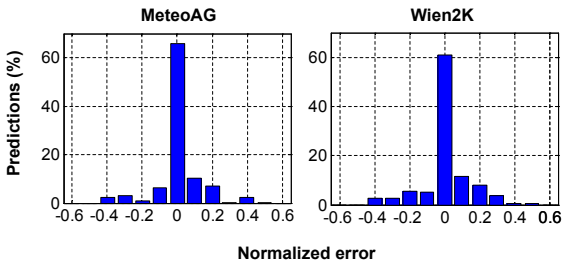


Figure 5: Relative error distribution of the two scientific workflows.

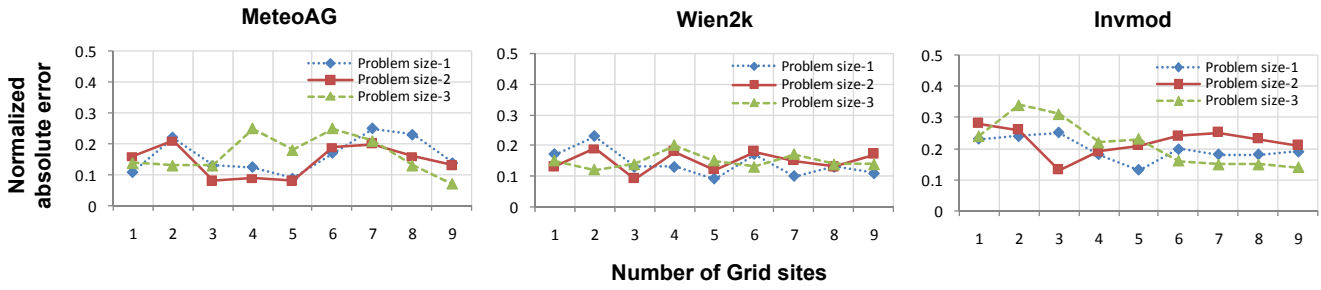


Figure 6: Normalized absolute error of predictions for the three workflows for different problem sizes.

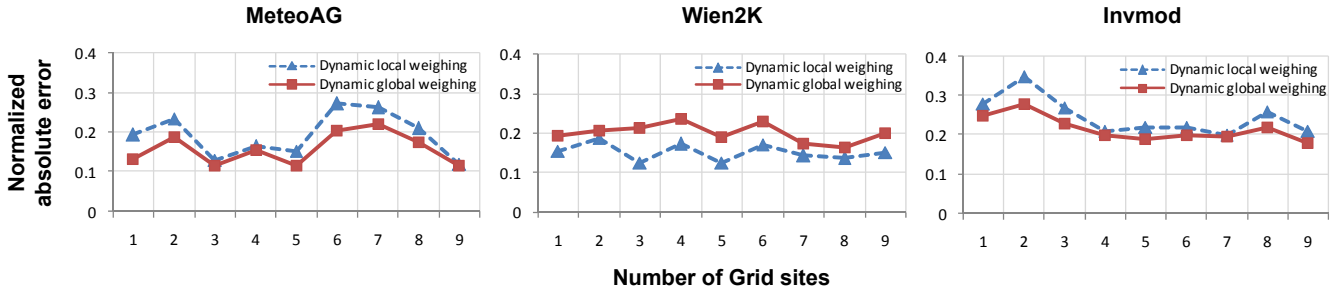


Figure 7: Normalized absolute error of predictions using *Global* and *Local weighing* for the three workflows.

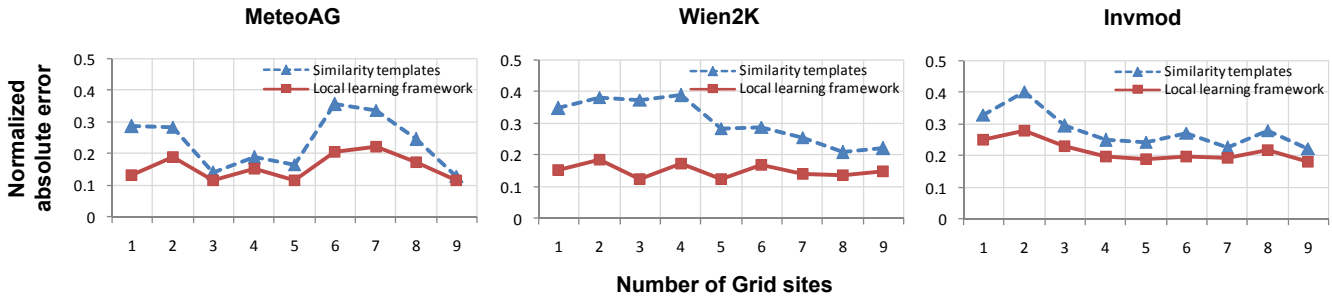


Figure 8: Normalized absolute error of predictions using *similarity templates* and *LLF* for the three workflows.

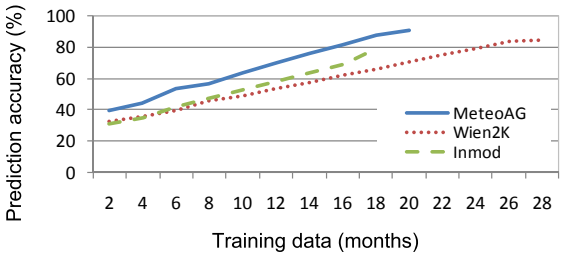


Figure 9: Accuracy of predictions for MeteoAG, Wien2K, and Invmod workflows for different amounts of training data.

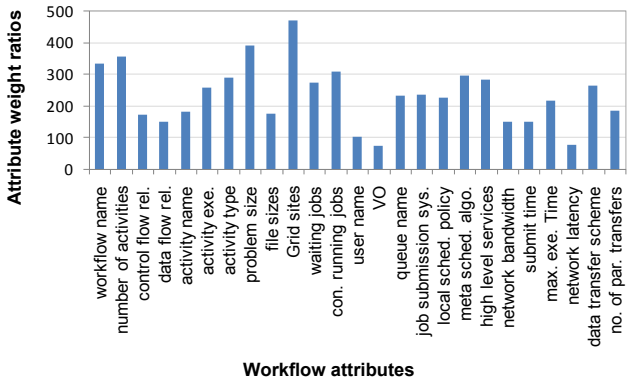


Figure 10: Ratios of weights assigned to different workflow attributes.

sites, *problem size* and *number of activities* are the attributes which highly affect the workflow execution time. Figure 10 shows ratios of attribute weights determined through evolution programming. A higher ratio of attribute weights indicates a higher effect on the workflow execution time.

In the course of our assessment of our prediction approach, we evaluated how effectively our predictions can guide the selection of optimization strategies. In the first evaluation,

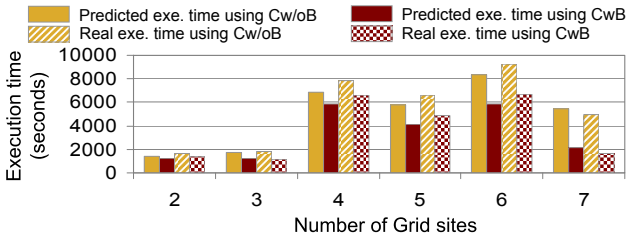


Figure 11: Measured and predicted execution time comparison of MeteoAG using two communication optimizations strategies: *communication with blocks* (CwB) and *without blocks* (Cw/oB).

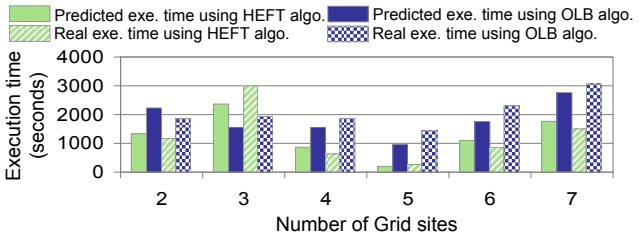


Figure 12: Measured and predicted execution time comparison of Wien2K with two scheduling algorithms: *HEFT* and *Opportunistic Load Balancing* (OLB).

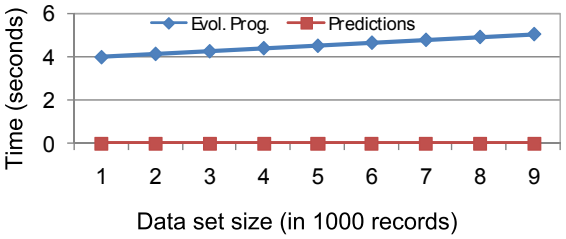


Figure 13: System overheads (average).

the experiments were designed to analyze comparison of executions of MeteoAG workflow using two communication optimizations (*communication based on blocks* and *without blocks* [18]), through measured and predicted execution times. For the block-based communication, a data collection C is partitioned into equal sized, contiguous blocks and each block is distributed onto a different loop iteration. Whereas, for communication without blocks the complete data collection C is distributed to the loops iterations [18]. To cover a range to data transfers (from small to large) during the workflow executions, the experiments were conducted to run the workflow on different number of Grid sites, from 2 to 7. Figure 11 shows comparison of the workflow executions using the two communication optimization strategies, in terms of measured and predicted execution times. The measured (real) execution times indicate that using block-based communication reduces the workflow execution time. The execution times predicted through our approach report exactly the same ranking (showing which strategy is better) of the two communication strategies.

In a second evaluation, we designed experiments to analyze the comparison of two well known meta scheduling algorithms (*Heterogeneous Earliest Finish Time (HEFT)* and *opportunistic load balancing*) for mapping the activities of Wien2K on different Grid sites from 2 from 7. Figure 12 shows this comparison for measured and predicted execution times. The comparison of measured workflow execution times shows that *opportunistic load balancing* executes workflow more efficiently (less execution time) when workflow is executed on two Grid sites, whereas, the *HEFT* algorithm performs better for other number of Grid sites (1, 3, 4, 5, 6, 7). The Workflow execution times predicted from our approach also show exactly the same ranking (showing which algorithm is better) of the two scheduling algorithms.

From both of our evaluations of comparison of workflow executions employing different optimizations strategies, we observed that the our predictions validate the measured ranking of different optimization strategies. Thus, we believe that despite of the small inaccuracies, our predictions can effectively guide the workflow runtime environment for the selection of different optimization strategies.

Based on our experiments we determined that the *k-weighted average* local model supersedes other local models. The $k-NN$ model did not effectively adjust the distant values. Particularly, we found that the $1-NN$ model adhered to local minima. The locally weighted linear regression did not perform better due to its ineffectiveness of modeling large number of workflow attributes.

To accelerate the NN search and reduce the distance computation cost we arrange the data set in a *k-d Tree* [1]. The Figure 13 shows average overheads of our workflow execution time prediction system (in terms of the time to find and optimize the best weights for workflow attributes and the time to predict workflow execution time) when run on an Intel Core2Duo 2.1 GHz machine with 3 GB of memory. The prediction time (learning for one instance of prediction query) ranged from 125 ms to 275 ms for data sizes ranging from 1000 to 9000. The corresponding evolution programming lapsed over 4 to 5 seconds.

7. RELATED WORK

There has been a series of related work for predicting execution time of single workflow activities, like [23, 30, 14, 11, 20], but to the best of our knowledge, the only effort to predict full workflow execution time (considering dependencies between workflow activities, distributed execution of workflow activities over multiple Grid sites, and effects of Grid middleware services) is our previous work [13] exploiting similarity templates. Though the similarity templates consider all of the workflow attributes of this paper, there are some weaknesses in the method based on similarity templates for predicting workflow execution time, which tends to reduce the prediction accuracy. First, similarity templates can not assign different priorities to workflow attributes w.r.t. their impact on the workflow execution time. Second, some of the important similarity information is lost by ignoring the *template classes* [13] containing fewer data instances. Third, even the best template does not contain all of the workflow attributes. This results in a potential loss of some similarity information. To overcome these weaknesses of similarity templates, in this paper, we introduce a totally new approach based on *LL*. Based on the difference in prediction accuracy of the two approaches as shown in Figure 8, we found that our new approach presented in this paper,

clearly outperforms the previous approach presented in [13]. We believe that our approach based on *LL* provides better prediction accuracy because *LL* considers a larger number of attributes for workflow similarity, and the dynamic weighing of the workflow attributes. In addition, the dynamic selection of NNs through *distance classes*, as compared to a fixed number of NNs used by other approaches [11] employing similar frameworks, also improves the accuracy. Moreover, the current approach, in contrast to the work in [13], also provides a confidence information as well as minimum and maximum execution time predictions.

Similarity of execution of single activities has been defined in [20, 11] using simple activity attributes. Lee et al. [11] have also described attributes to define similarity of resource states and policies, and employed instance based learning to predict execution time of single activities. In contrast to these efforts which are for execution time predictions of single activities on single Grid sites, our work is for execution time prediction of full workflows which comprise multiple activities with complex dependencies among them. Furthermore, our work targets execution of workflows in a distributed fashion on multiple Grid sites. We consider all major workflow attributes that describe workflow execution at different Grid levels of Grid infrastructures (like Grid site, network, etc.). In particular the workflow structure attributes are also included to consider workflow structure similarities. Another major difference in our approach from the existing approaches (for activity execution time prediction) is the inclusion of problem size attribute to describe a workflow execution. This enable us for a more precise definition of workflow execution similarity and thus gain higher prediction accuracies. Similar to our work, Lee et al. also have exploited bias-variance tradeoff in [11], but their criteria of tradeoff is different from ours.

Glatard et al. [8] use probabilistic models to analyze Grid workflow performance by considering execution times of individual activities and data transfers among the activities, and modeling the rest of time (taken in different execution phases) as a random variable. Gelenbe et al [7] and Mussi et al. [12] also considered the execution time of a task graph as a random variable and determined its distribution based on graph parameters. All these work ([8, 7, 12]) assume basic workflows, ignoring complex control flows among activities (loops over different (sets of) activities, conditional executions, etc), and variations in execution time due to the lack of modeling input data. It is noteworthy that the number of times a workflow activity is executed depends often on input data, which is of core importance for overall workflow execution time. Our approach considers the workflow structure attributes, the problem size, and the effects of different optimizations performed by different high level middleware services (i.e. scheduling algorithms, different data transfers schemes etc.).

8. CONCLUSIONS AND FUTURE WORK

Workflow execution time predictions are of critical importance to understand the execution behavior and guide the optimization of Grid workflows. Only little research has been done to explore this topic so far due to the complexity of workflows and due to dynamic behavior of the Grid. In this paper, we introduced a novel method for estimating the execution time of Grid workflows which is based on a local learning framework. The contributions of the presented work are as follows. Based on our experiments with three real-world workflows on a real Grid infrastructure, we believe that our new approach outperforms previous work. The effect of different attributes on workflow performance is evaluated in terms of attribute weights that are determined from the workflow traces. The dynamic weighing scheme optimizes these weights. The dynamic selection of local data through distance classes further improves the prediction accuracy. Likewise, we introduce prediction intervals, with high accuracy for predicting the maximum and the minimum workflow execution time. We observed that the problem size, Grid sites and meta scheduling algorithm have the highest effects on workflow performance. We also demonstrated that our prediction approach has a promising potential to guide the application of optimization strategies of a Grid runtime environment by providing workflow execution time predictions. Our prediction systems has been implemented as a part of the ASKALON Grid workflow runtime environment and validated for against three real-world workflow applications.

In future work, we want to evaluate our approach with more data of our considered workflows. Moreover, we intend to evaluate our approach for additional real world workflows.

9. REFERENCES

[1] C. Atkeson, A. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, April 1997.

[2] K. Deb. *Multi-objective Optimization Using Evolutionary Algorithms*. John Wiley and Sons, 2nd edition, 2002.

[3] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming Journal*, 13(3):219–237, 2005.

[4] Enabling Grids for E-science. www.eu-egee.org/.

[5] T. Fahringer, R. Prodan, R. Duan, J. Hofer, F. Nadeem, F. Nerieri, S. Podlipnig, J. Qin, M. Siddiqui, H. L. Truong, A. Villazon, and M. Wieczorek. *Scientific Workflows for Grids*, chapter ASKALON: A Development and Grid Computing Environment for Scientific Workflows. Springer Verlag, 2007.

[6] D. G. Feitelson and B. Nitzberg. Job characteristics of a production parallel scientific workload on the NASA ames iPSC/860. In *workshop on job scheduling strategies for parallel processing*, Santa Barbara, USA, 1995.

[7] E. Gelenbe, E. Montagne, R. Suros, and C. M. Woodside. A performance model of block structured parallel programs. In *Proceedings of the international workshop on Parallel algorithms & architectures*, pages 127–138, Amsterdam, The Netherlands, The Netherlands, 1986. North-Holland Publishing Co.

[8] T. Glatard, J. Montagnat, and X. Pennec. A probabilistic model to analyse workflow performance on production grids. In *CCGRID '08: Proceedings of the Eighth IEEE International Symposium on Cluster Computing and the Grid*, pages 510–517, Lyon, France, 2008. IEEE Computer Society.

[9] T. Haluk, H. Salim, and W. Min-you. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distribributed Systems*, 13(3):260–274, 2002.

[10] S. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, 3rd edition, 2008.

[11] H. Li, J. Chen, Y. Tao, D. Gro, and L. Wolters. Improving a local learning technique for queuewait time predictions. In *CCGRID '06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, pages 335–342, Singapore, 16-19 May 2006. IEEE Computer Society.

[12] P. Mussi and P. Nain. Evaluation of parallel execution of program tree structures. *SIGMETRICS Performance Evaluation Review*, 12(3):78–87, 1984.

[13] F. Nadeem and T. Fahringer. Similarity templates based distributed application workflow performance prediction. In *Proc. of the Eighth IEEE International Symposium on Cluster Computing and the Grid*, Shanghai, China, May 18-21 2009.

[14] F. Nadeem, M. M. Yousaf, R. Prodan, and T. Fahringer. Soft benchmarks-based application performance prediction using a minimum training set. In *E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*, page 71, Amsterdam, The Netherlands, 2006. IEEE Computer Society.

[15] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, Nov. 2004.

[16] P. Blaha et al. *WIEN2k: An Augmented Plane Wave plus Local Orbitals Program for Calculating Crystal Properties*. Institute of Physical and Theoretical Chemistry, TU Vienna, 2001.

[17] Platform Computing. Load sharing facility (lsf). www.platform.com.

[18] J. Qin and T. Fahringer. Advanced data flow support for scientific grid workflow applications. In *ACM/IEEE conference on Supercomputing*, Reno, USA, 2007.

[19] F. Schüller, J. Qin, F. Nadeem, R. Prodan, T. Fahringer, and G. Mayr. Performance, Scalability and Quality of the Meteorological Grid Workflow. In *2nd Austrian Grid Symposium, Innsbruck, Austria*. OCG Verlag, Sep. 2006.

[20] W. Smith, I. Foster, and V. Taylor. Predicting application run times with historical information. *J. Parallel Distrib. Comput.*, 64(9):1007–1016, 2004.

[21] Sun Microsystems. Sun grid engine (sge). www.sun.com/software/sge/.

[22] I. Taylor, I. Wang, M. Shields, and S. Majithia. Distributed computing with triana on the grid. *Concurrency and Computation: Practice & Experience*, 17(9):1197–1214, 2005.

[23] V. Taylor, X. Wu, J. Geisler, and R. Stevens. Using kernel couplings to predict parallel application performance. In *HPDC '02: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, page 125, Washington, DC, USA, 2002. IEEE Computer Society.

[24] The Austrian Grid Consortium. www.austriangrid.at.

[25] The Grid5000. www.grid5000.fr.

[26] D. Theiner and M. Wieczorek. Reduction of calibration time of distributed hydrological models by use of grid computing and nonlinear optimisation algorithms. In *7th International Conference on Hydroinformatics (HIC 2006)*, Acropolis, Nice, France, September 2006 2006.

[27] D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.

[28] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.

[29] A. Wombacher and M. Rozie. Piloting an empirical study on measures for workflow similarity. In *IEEE International Conference on Services Computing*, Chicago, USA, 2006.

[30] E. J. H. Yero and M. A. A. Henriques. Contention-sensitive static performance prediction for parallel distributed applications. *Perform. Eval.*, 63(4):265–277, 2006.