$FReeP$: towards parameter recommendation in scientific workflows using preference learning*

Daniel Silva Jr.[1], **Aline Paes**[1], **Esther Pacitti**[2], **Daniel de Oliveira**[1]

[1]Department of Computer Science, Universidade Federal Fluminense - Brazil

danieljunior@id.uff.br, {alinepaes,danielcmo}@ic.uff.br

[2]Inria, LIRMM and University of Montpellier - France

Esther.Pacitti@lirmm.fr

Abstract. *Scientific workflows are a de facto standard for modeling scientific experiments. However, several workflows have too many parameters to be manually configured. Poor choices of parameter values may lead to unsuccessful executions of the workflow. In this paper, we present $FReeP$, a parameter recommendation algorithm that suggests a value to a parameter that agrees with the user preferences. $FReeP$ is based on the Preference Learning technique. A preliminary experimental evaluation performed over the SciPhy workflow showed the feasibility of $FReeP$ to recommend parameter values for scientific workflows.*

1. Introduction

Scientific workflows are considered a *de facto* standard for modeling scientific experiments that are compute- and data-intensive [Zhao et al. 2008]. They are abstractions that represent the flow of data among activities (*i.e.,* program invocations). The Scientific Workflow Management Systems (SWfMS) are responsible for managing the execution of workflows and collecting provenance data [Freire et al. 2008], which represent the execution history of the workflow. A workflow can be formally defined as a directed acyclic graph $W(A, Dep)$. The nodes $A = \{a_1, a_2, ..., a_n\}$ are the activities and the edges Dep represent the data dependencies among activities in A. Thus, given $a_i \mid (1 \leq i \leq n)$, the set $P = \{p_1, p_2, .., p_m\}$ represents the possible input parameters for activity a_i that define the behavior of a_i.

Scientific Workflows are applied in many fields, such as biology and astronomy. Given the increasingly complexity of experiments in these domains [Zhao et al. 2008], many workflows have many parameters to be configured by users (*e.g.,* > 40). The configuration of such parameters is a sensitive point because it can impact the execution time of the workflow and the usefulness of the produced results. This way, it is required that the user is able to configure the workflow (*e.g.,* setting values for all the m parameters of an activity a_i) as best as possible. Let us take as an example the workflow SciPhy [Ocaña et al. 2011]. SciPhy aims at generating phylogenetic trees (*i.e.,* trees that represent the evolutionary history of an organism). It is composed of four activities: (i) sequence alignment; (ii) conversion of alignment format; (iii) search for the best evolutionary model; and (iv) construction of the phylogenetic tree. Although conceptually simple (*i.e.,* it has only four activities), SciPhy can be complex to be configured because of the number of parameters that should be explored. In addition, the user does not necessarily know *a priori* which configuration generates the best-quality phylogenetic tree. For example, each input file that contains DNA or RNA sequences has a number

*Authors would like to thank FAPERJ, CAPES and CNPq for partially sponsoring this research

of sequences num_seq and a maximum length of sequence $length\_seq$, which are input parameters of activity (i). However, to activity (iv) the input parameter $bootstrap\_replicate$ has its utility limited depending on the choice of values for $length\_seq$. A poor choice of such values can generate a worthless result (in addition to the loss of time and resources).

Thus, it is interesting that the parameter values can be *automatically* recommended to users by a Recommendation System (RS) to increase the chances of producing good results. For example, if in a given SciPhy execution the user sets the value of $num\_seq = 100$, it would be better that the value of $bootstrap\_replicate$ to be recommended by an RS followed the value of num_seq in order to avoid incompatibility of the parameter values. This type of recommendation is feasible, since the SWfMSs collect provenance data, but it is not simple to be performed. Provenance can be used for future recommendations as we know which executions produced successful results and which parameter values were used. The motivation of this paper is, therefore, to recommend parameter values of workflow activities using provenance data collected in previous workflow executions. Thus, we propose a parameter recommendation algorithm named $FReeP$ for scientific workflows that benefits from the other chosen parameters. This way, $FReeP$ is able to suggest a value to a set of parameters that agrees with the user preferences. To accommodate such preferences together with the more appropriate recommendation, we follow a Preference Learning [Fürnkranz and Hüllermeier 2011] technique. The experimental evaluation performed with the SciPhy workflow showed the feasibility of $FReeP$ to recommend parameter values.

2. A Brief on Preference Learning

Recommendation algorithms aim at suggesting the most relevant items to solve a task that requires a choice. In a *personalized* recommendation, each user receives his/her own list of items, based on his/her *preferences*. From the Artificial Intelligence point of view, a preference is an expression of the problem's constraints, but allowing some sort of relaxation [Fürnkranz and Hüllermeier 2011]. In general, Preference Learning consists of inducing a predictive function that, given a set of already established-as-preferred items, it predicts the preferences for a new set of items. The most likely research task in this area is "Learning how to rank", rising from the need of obtaining an ordering relation among the preferences. The ordering task may focus on the class label, directly on the instances, or on a subsect of the objects. Particularly, one of the most used technique to learn preferences is *Pairwise Label Ranking* [Hüllermeier et al. 2008]. It consists of learning the preferences by decomposing the problem in smaller binary preferences problems, *i.e.*, preferences between pairs of classes. Next, an ordering is induced relying on methods that minimize the loss function. This function, in turn, is computed according to the preferences that are violated, considering each different combination of classes.

3. FReeP: Feature Recommender from Preferences

In this paper, we propose a recommendation algorithm named $FReeP$ that relies on provenance data, preference learning, and voting systems to recommend a value for a specified parameter. The recommendation provided by $FReeP$ is the combination of a set of selected recommendations. These recommendations are created according to other specified parameter values. $FReeP$ belongs to the category of Collaborative Filtering [Herlocker et al. 2004] methods and is presented in Algorithm 1.

Algorithm 1 FReeP

Require:

S: { $(param_1^1, val_1^1)$,...$(param_l^m, val_l^m)$ } ▷ l is the number of workflow parameters and m is the number of tuples in the provenance database
F : {attr | attr is a workflow parameter }
C : {attr_preference | attr_preference ∈ F} ▷ *attr_preference* is a workflow parameter where the user defined a value
f(attr) : {preference_value | attr ∈ C} ▷ *preference_value* is the value defined by user for parameter *attr*
P : {(attr,attr_preference) | attr ∈ C ∧ attr_preference ∈ f(attr)} y | y ∈ (F - C)
procedure FREEP(type) ▷ Type is used to select between pure KNN or Label Rank
    $votes \leftarrow \emptyset$; $FS \leftarrow$ POWER_SET$(C) \setminus \emptyset$ ▷ POWERSET it the math operation that returns the set of all subsets
    for each $set \in FS$ **do**
        $horizontal\_partition \leftarrow \emptyset$ ▷ Horizontal partition holds only the instances matching the user preferences
        for each $tuple \in S$ **do** ▷ $tuple$ refers to each tuple in the provenance database
            $flag \leftarrow true$
            for each $(attr, value) \in tuple$ **do**
                if $attr \in C \ \& \ (attr, value) \notin P$ **then**
                    $flag \leftarrow false$
            if $flag$ **then**
                $horizontal\_partition \leftarrow horizontal\_partition \cup \{tuple\}$
        $set \leftarrow set \cup \{y\}$; $vertical\_partition \leftarrow \emptyset$
        for each $tuple \in horizontal\_partition$ **do**
            $filtered\_record\_columns \leftarrow \emptyset$
            for each $(attr, value) \in tuple$ **do**
                if $attr \in set$ **then**
                    $filtered\_record\_columns \leftarrow filtered\_record\_columns \cup (attr, value)$
            $vertical\_partition \leftarrow vertical\_partition \cup filtered\_record\_columns$
        $recommender \leftarrow$ SELECT_RECOMM$(type, vertical_partition)$ ▷ Builds the pure KNN or the KNN + LabelRank
        $to\_recommend \leftarrow \{attr\_preference \mid (attr, attr\_preference) \in P, attr \in set\}$
        $vote \leftarrow$ RECOMMEND$(to_recommend, recommender, type)$ ▷ KNN returns the predicted value, while LabelRank returns the results of the ranking process
        $votes \leftarrow votes \cup \{vote\}$
    $recomendation \leftarrow$ GET_RECOMENDATION$(type, votes)$
    return $recomendation$
function GET_RECOMMENDATION($type$, $votes$)
    if $type ==' KNN'$ **then**
        return ARGMAXCOUNT$(votes)$
    else
        return BORDACOUNT$(votes)$
function BORDACOUNT($rankings$)
    $labels \leftarrow \emptyset$; $labels\_votes \leftarrow \emptyset$
    for each $rank \in rankings$ **do**
        for each $label \in rank$ **do**
            if $label \notin labels$ **then**
                $labels \leftarrow labels \cup label$
                $labels\_votes \leftarrow labels\_votes \cup (label, 0)$
    for each $rank \in rankings$ **do**
        $weight \leftarrow$ LENGTH$(rank) - 1$
        for each $label \in rank$ **do**
            $total\_votes \leftarrow total\_votes \mid (vote, total\_votes) \in labels\_votes, vote == label$
            $new\_total\_votes \leftarrow total\_votes + 2^{weight}$
            $label\_vote \leftarrow (label, new\_total\_votes)$
            $weight \leftarrow weight - 1$
    $recomendation \leftarrow$ ARGMAXLABEL$(labels_votes)$
    return $recomendation$

To perform the parameter recommendation we rely on the tuples extracted from the Provenance database S. These tuples represent successful executions of a given workflow. The set of preferences P, *i.e.,* the parameters for which the scientist already has a set of values, and the parameter to be recommended y are also provided as input to $FReeP$. The first step is to create the set of all subsets of the instances (the power set)[1], grounded by the preferred parameters, called FS. For each subset of FS a horizontal partition is performed over the provenance data, by selecting only the tuples where each parameter (*i.e.,* attribute) has the same value as specified in the user preferences. Next, only the attributes of the tuples that represent the parameters present in the user preferences are gathered (*i.e.,* vertical partition). After that, the algorithm may follow two paths: either it uses a *KNN* classifier [Cover and Hart 1967], or a *LabelRank* [Hüllermeier et al. 2008] classifier. Both of them use the aforementioned partitions to make the recommendation for the parameter y. If the choice is *KNN*, the prediction model is trained with the data in the partition and their respective values for the y parameter, yielding the prediction of values for the other parameter in the partition and stated in the user preferences. On the other hand, if the choice is *LabelRank*, a ranking function is trained in the same way as the *KNN*, however, it returns a sorted list, from the most appropriate value for y to the least one. In this paper, we choose to use the *Pairwise Label Ranking* method to approximate the ranking function.

Each partition generates a possibly different recommendation to the y parameter. This implies in combining the different recommendations in order to arrive at a consensus of what it is the best one. When the chosen method is *KNN*, the recommendation for each partition is a single value, resulting in a list of recommended values. In this case, we use the Simple Voting System [Fishburn 1974] where the value that is most cited among the recommendations is the selected one. On the other hand, the *LabelRank* method returns the recommendations as a *ranking*. The result after the interaction at each partition is a list of *rankings*. Here, we could also employ a simple voting system based on the value ranked as the first one. However, this would neglect other well-ranked values among the different partitions. Thus, we use the Border Count method [Black 1976] that combines all rankings into one, after giving scores for each parameter value according to its position in each ranking. After that, we use the highest ranking value as the final recommendation.

4. Experimental Results

To evaluate $FReeP$, we used the provenance data collected by SciCumulus SWfMS when executing SciPhy workflow [Ocaña et al. 2011]. SciPhy was developed to build phylogenetic trees from DNA, RNA and amino acid sequences. The dataset used to train de Machine Learning models is composed of 376 examples of executions that have not ended at a failure. We follow a 5-Fold Cross Validation procedure [Refaeilzadeh et al. 2016], dividing the examples into 5 disjoint sets, and, at each iteration, 4 sets were used to train the models, while the remaining set was used as test. We considered $K \in [3, 5, 7]$ in both pure KNN and Label Rank methods. We experiment with the recommendation of each parameter of the workflow, separately. We compute the accuracy of both models to evaluate the capability of both approaches in suggesting the right value parameter. Figure 4 shows the experimental results. We can see a clear difference in the accuracies between the recommendation for the parameter num_aligns and all the others. While for the first the recommendation reached values greater than 90%, the others reached accuracies varying from 40% to 60%. This can be explained

[1]In this initial version, we follow the most basic way of choosing the subsets by using the powerset of the original data. This may lead to an exponential number of partitions.

by the smaller variation of values for the parameter num_aligns in the input dataset. Different values of the K parameter did not lead to significant changes in the recommendations, with low standard deviation among the folds.

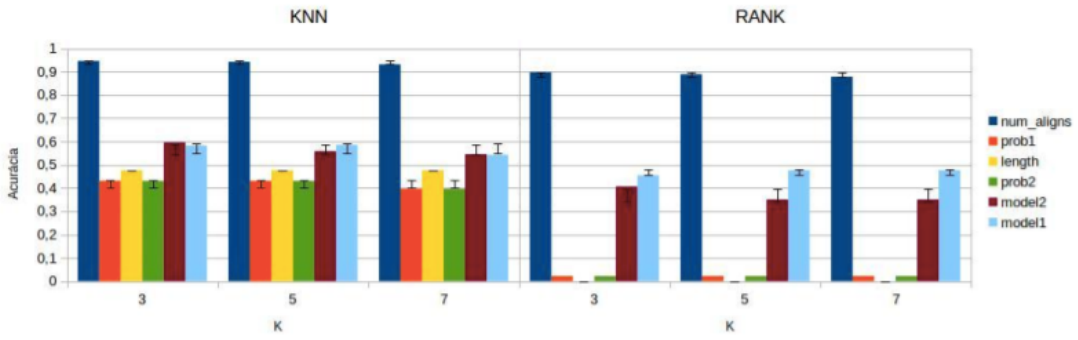

Figura 1. Accuracy Results for both Pure KNN and LabelRank Methods

Furthermore, the rank-based model obtained worse results than KNN. Although they both come close regarding the parameters num_aligns, $model\_1$, and $model\_2$, the rank accuracy results for the rest of the parameters are close to zero. One explanation for such behavior is the numerical nature of these attributes and their very sparse values in the input dataset, which produces very different rankings along the training.

5. Related Work

The ranking strategy used in this paper was also used in the recommendation context but applied to movies scenario [Pessiot et al. 2007]. The voting method has been used to decrease the training time with large datasets in a movies recommendation task, without deteriorating the quality of the recommendation [Das et al. 2014, Mukherjee et al. 2003]. Particularly, in [Mukherjee et al. 2003], Preference Learning is also used to leverage the recommendation. Regarding recommendation in workflows, Halioui [Halioui et al. 2016] combined natural language processing with ontologies to recommend searching keywords. In [Soomro et al. 2015], a pattern-based recommendation approach was built to suggest workflows composition. [Cheng et al. 2015] propose an approach for identifying and recommending the workflows for reference using semantic similarity. However, the aforementioned approaches do not recommend values to the parameters.

6. Conclusions and Future Work

The effective use of scientific workflows and SWfMS have fostered the scientific experimentation and its analysis. However, several experiments modeled as workflows have a large set of parameters to be configured, which is not a trivial task to accomplish. While in some cases the scientist may be working on an experiment where he/she already knows at least a subset of the most appropriate values, he/she may not know which is the best values to assign to the others parameters of the workflow. In addition, a poor choice of parameters may lead to undesired results and loss of time. In this paper, we propose a parameter recommendation algorithm called $FReeP$, based on Preference Learning and Voting Systems to recommend values for parameters, while restraining the recommendations to the user preferences. Experiments showed that when we employ a *KNN* classifier we can reach the correct parameter value in most cases, even in the presence of a reduced training set. On the other hand, we

still need to improve the methods based on *LabelRank* so that it can achieve its full potential. As future work, we plan: (i) to consider more clever partition strategies to make the training more efficient; (ii) to rely on a non-uniform choice from the KNN classifier; (iii) to explore other classifiers; and (iv) to test other voting schemas.

Referências

Black, D. (1976). Partial justification of the borda count. *Public Choice*, 28(1):1–15.

Cheng, Z., Zhou, Z., and Wang, X. (2015). Scientific workflow clustering and recommendation. In *11th International Conf. on Semantics, Knowledge and Grids (SKG)*, pages 272–274.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.

Das, J., Mukherjee, P., Majumder, S., and Gupta, P. (2014). Clustering-based recommender system using principles of voting theory. In *IC3I*, pages 230–235. IEEE.

Fishburn, P. C. (1974). Simple voting systems and majority rule. *Systems Research and Behavioral Science*, 19(3):166–176.

Freire, J., Koop, D., Santos, E., and Silva, C. T. (2008). Provenance for Computational Tasks: A Survey. *Computing in Science & Engineering*, pages 20–30.

Fürnkranz, J. and Hüllermeier, E. (2011). Preference learning. In *Encyclopedia of Machine Learning*, pages 789–795. Springer.

Halioui, A., Valtchev, P., and Diallo, A. B. (2016). Towards an ontology-based recommender system for relevant bioinformatics workflows. *bioRxiv*, page 082776.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53.

Hüllermeier, E., Fürnkranz, J., Cheng, W., and Brinker, K. (2008). Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16-17):1897–1916.

Mukherjee, R., Sajja, N., and Sen, S. (2003). A movie recommendation system–an application of voting theory in user modeling. *User Modeling and User-Adapted Interaction*, 13(1-2):5–33.

Ocaña, K. A., de Oliveira, D., Ogasawara, E., Dávila, A. M., Lima, A. A., and Mattoso, M. (2011). Sciphy: a cloud-based workflow for phylogenetic analysis of drug targets in protozoan genomes. In *BSB11*, pages 66–70. Springer.

Pessiot, J., Truong, T., Usunier, N., Amini, M., and Gallinari, P. (2007). Learning to rank for collaborative filtering. In *ICEIS 2007 - Proc. of the 9th International Conf. on Enterprise Information Systems*, pages 145–151.

Refaeilzadeh, P., Tang, L., and Liu, H. (2016). Cross-validation. *Encyclopedia of database systems*, pages 1–7.

Soomro, K., Munir, K., and McClatchey, R. (2015). Incorporating semantics in pattern-based scientific workflow recommender systems: Improving the accuracy of recommendations. In *SAI'2015*, pages 565–571. IEEE.

Zhao, Y., Raicu, I., and Foster, I. (2008). Scientific workflow systems for 21st century, new bottle or new wine? In *IEEE Services*, pages 467–471. IEEE.